NEC Personal Computer

PC-8801mkII MR
N_{88}-BASIC/N_{88}-日本語BASIC
GUIDE BOOK



PC-8801MR-GB

ご注意

(1)本書の内容の一部または全部を無断転載することは
　禁止されています．
(2)本書の内容に関しては将来予告なしに変更することがあります．
(3)本書は内容について万全を期して作成いたしましたが，
　万一ご不審な点や誤り，記載もれなど
　お気付きのことがありましたらご連絡ください．
(4)運用した結果の影響については(3)項にかかわらず
　責任を負いかねますのでご了承ください．
(5)乱丁，落丁はお取り替えいたします．

PC-8801 MKⅡMR
N88-BASIC, N88-日本語BASICガイドブック

もくじ

付　録

資　料

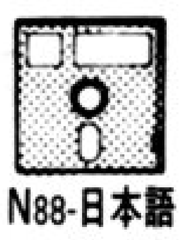
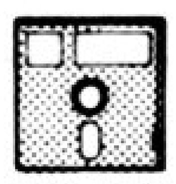
は じ め に

このマニュアルは，N88-BASICとN88-日本語BASICを使ってプログラムをされる方のためのガイドブックとして編集したものです．
BASICの各々の命令や関数は，BASICリファレンスマニュアルにアルファベット順に説明されていますが，このガイドブックではプログラムの目的別に必要な命令や関数を組み合わせて，解説を行います．
N88-BASICとN88-日本語BASICとは，基本的には同じように使用しますが，N88-日本語BASICでは，その名前のとおり，日本語の処理が強化されています．ガイドブックでは，N88-日本語BASICをご使用になる方のために，第2章 日本語文字の入力と第3章 日本語処理の2つの章を設けてあります．
それ以外の章は，N88-BASIC，N88-日本語BASICで共通です．
BASICには機械語モニタと，ターミナルモードに移行するためのMONコマンドと，TERMコマンドを持っています．ガイドブックでは，付録でこれらのモードの使い方を説明します．また，機械語でシンセサイザICを操作する場合の資料も付録3に掲載いたしました．
BASICガイドブックを参考にしていただいたことによって，少しでもよいプログラムを書いていただければ幸いです．

このマニュアルで使われている用語

PC-8801MKIIMRには強力な2つのBASIC
−N88-BASICとN88-日本語BASIC−
が装備されています.
N88-BASICはいくつかのモードを持ち,
また,ディスク装置を使う場合,使わない場合などでも
少しずつ機能が異なります.
ここでは,BASICのバージョンやモードを
中心として,このマニュアルで使われている用語を
まとめておきます.

1. BASICの種類やモード名

用　　語	意　　味	参照マニュアル，章，シンボル
N88-BASIC V1	従来のN88-BASIC（PC-8801, PC-8801MKII用のN88-BASIC）と互換性があるモード.PC-8801MKIIMRのBASICモードスイッチをN88V1にセットしてスタートさせる. N88-BASICシステムディスクを使ってスタートさせるV1 DISKと,ディスクを使わないV1 ROMの2つのモードの総称.	ユーザーズガイド 第6章 BASICリファレンスマニュアル 1.1参照
N88-BASIC V2	PC-8801MKIISR用の，N88-BASICの互換性のあるモード. PC-8801MKIIMRのBASICモードスイッチをN88V2にセットしてスタートさせる. N88-BASICシステムディスクを使ってスタートさせるV2 DISKと，ディスクを使わないV2 ROMの2つのモードの総称.	ユーザーズガイド 第6章 BASICリファレンスマニュアル 1.1 参照
N88-日本語BASIC	N88-BASIC V2に,日本語処理機能を付加して,PC-8801MKIIMR用に開発されたBASIC. PC-8801MKIIMRの，BASICモードスイッチをN88V2にセットし，N88-日本語BASICシステムディスクを使ってスタートさせる． 5.25インチディスクドライブに， システムディスクをセットした状態で動作する.	ユーザーズガイド 第6章 BASICリファレンスマニュアル 1.1 参照  N88-日本語
N88-BASIC DISK version	V1 DISKとV2 DISKの総称.  ディスク	ユーザーズガイド 第6章 BASICリファレンスマニュアル 1.1 参照
N88-BASIC ROM version	V1 ROMとV2 ROMの総称.	
BASIC	パソコンで一般的に広く使われているBASICすべての名称. 狭義には,PC-8801MKIIMRのN88-BASIC V1, V2, N88-日本語BASICを意味する.	
拡張命令	N88-BASIC V2およびN88-日本語BASICで，NEW CMDコマンドを実行することにより，使用可能になるサウンド機能やアナログパレット機能をサポートする命令群.	ユーザーズガイド 第6章，BASICリファレンスマニュアル 第4章参照.

用　　　語	意　　　　味	参照マニュアル，章，シンボル
タートルグラフィック拡張命令	N88-BASIC V1, V2で，機械語プログラム(ファイル名は@exst)をメモリ内にロードすることによって，使用可能になるタートルグラフィックス機能などをサポートする命令群．	ユーザーズガイド第6章，BASICリファレンスマニュアル第3章参照．

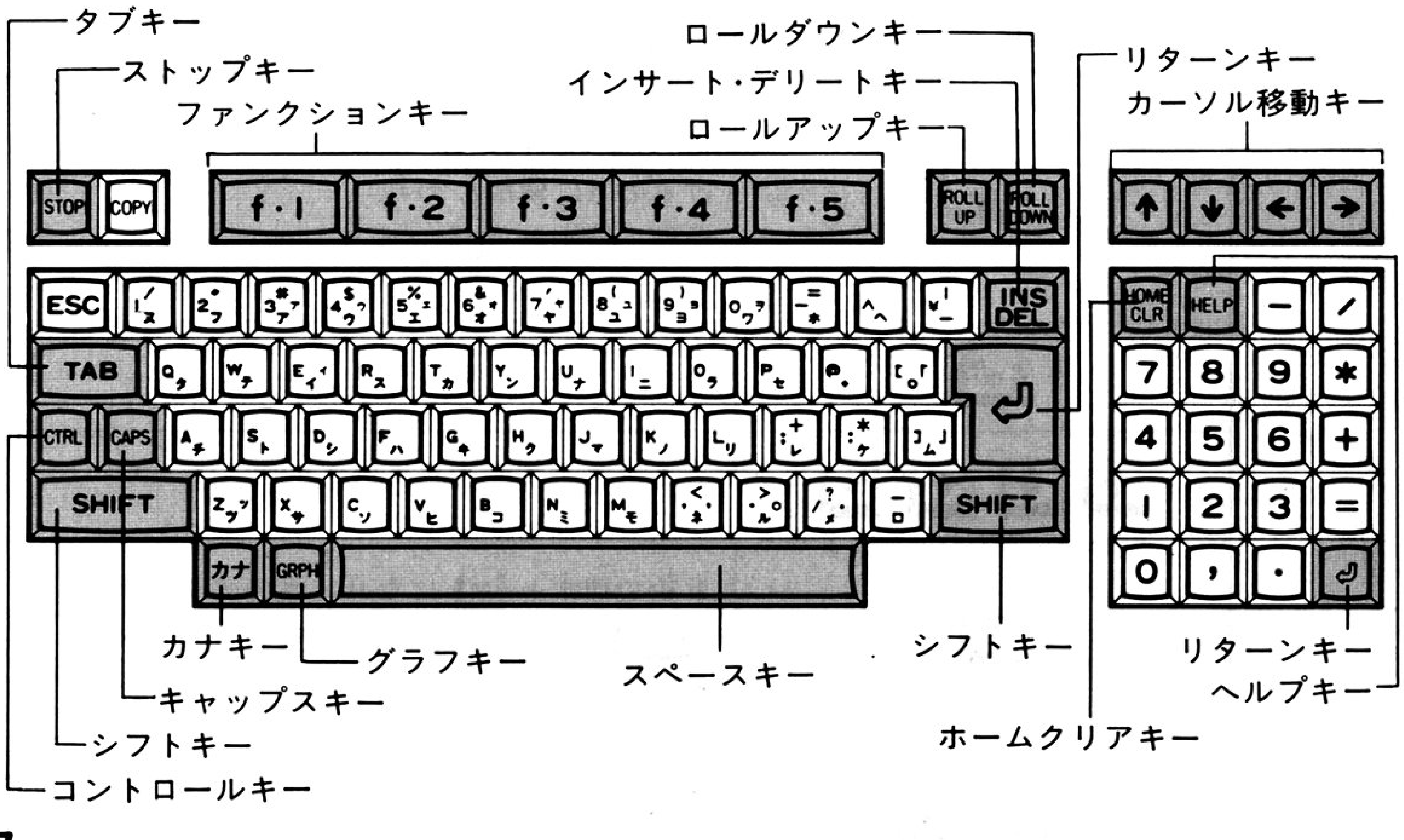
2. N88-日本語BASICで使われる用語

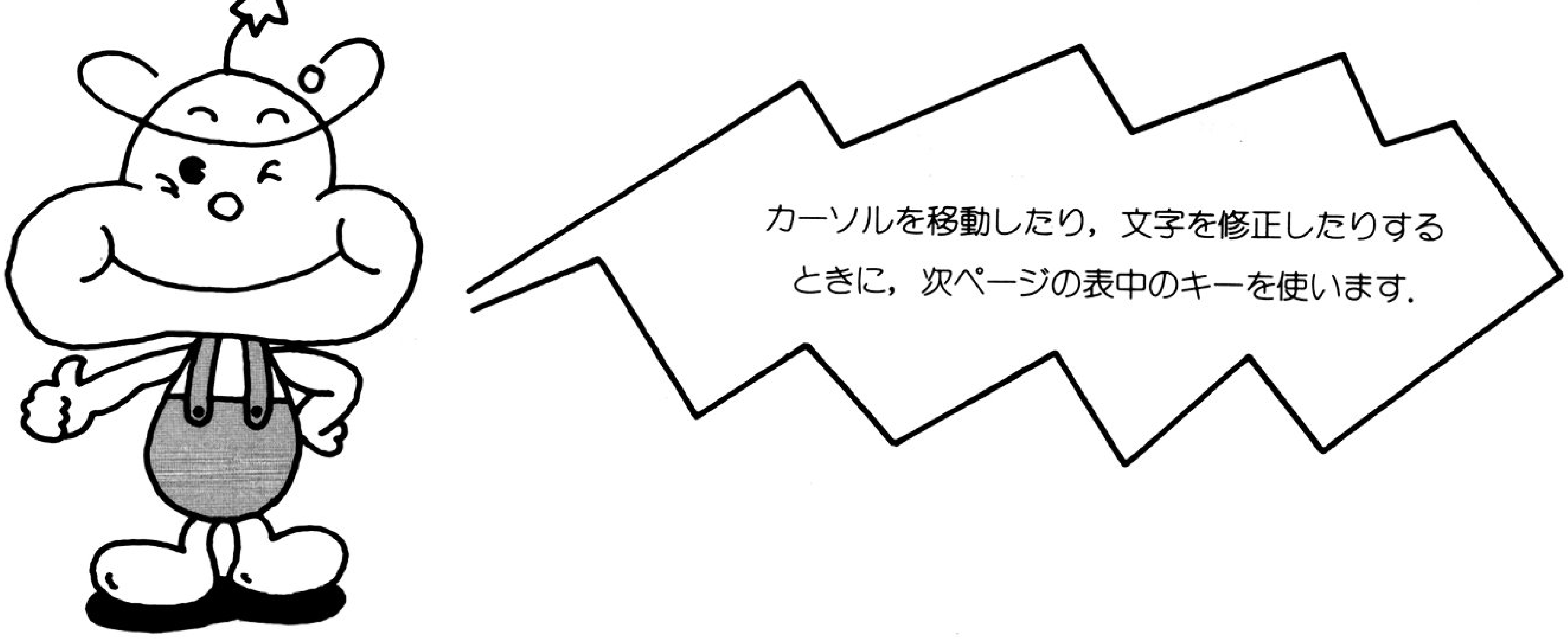
用　　　語	意　　　　味	参照マニュアル，章，シンボル
日本語モード	SCREEN文の第1パラメータを3または，4に指定したときのモード．全角文字の画面表示やキー入力が可能．N88-日本語BASICのスタート時は，標準ディスプレイ使用時はSCREEN3，専用ディスプレイ使用時は，SCREEN4に設定されている．	BASICガイドブック第3章 参照．
グラフィックシンボルモード	日本語モードに対するモードで，SCREEN文の第1パラメータを，0, 1, 2に指定したときのモード．全角文字の画面表示，キー入力はできない．全角文字に相当するコードを表示させると，グラフィックシンボル(資料6 参照)を含む半角文字が表示される．	BASICガイドブック第3章 参照．
全角文字	日本語を表すための文字(BASICリファレンスマニュアル資料8 日本語コードにあげられている文字)．BASICは，シフトJISコードによって全角文字を扱い，MID$などの関数は，全角文字1文字を半角文字2文字と等価に扱う．	BASICガイドブック第2章，第3章，BASICリファレンスマニュアル資料8 参照．
半角文字	英数字，カタカナ，コントロールコードを含む文字(資料5 キャラクタコードにあげられている256文字)．N88-BASICなどの日本語処理機能を持たないBASICでは，半角文字しか使用できない．	BASICガイドブック第2章，資料5 参照．
全角文字入力モード 半角文字入力モード	日本語モードでは，全角文字と半角文字とを混在して使用することができる．キー入力時には f・1 (全角/半角)によって，全角文字を入力するためのモード(全角文字入力モード)と，半角文字を入力するモード(半角文字入力モード)とを切り替えながら行う．	BASICガイドブック第2章 参照．

第1章 スクリーンエディタ

N88-BASIC/N88-日本語BASICでは，
プログラムを作ったり，コマンドやステートメントを
ダイレクトに実行させたりするときに
スクリーンエディタという画面編集機能を使います．
スクリーンエディタは，ディスプレイ画面上のどの位置にでも
カーソルを移動して，入力，修正をすることができます．
これによって修正する行を改めて最初から
入力する必要がなくなり，プログラムの
入力や修正も短時間で行うことができます．
この章では，スクリーンエディタの基本的な使い方を，
英数字だけを使って説明していきます．
N88-日本語BASICでの日本語文字を混じえた
スクリーンエディタの使い方は，第2章を
ご覧ください．

1．編集に使うキーの操作





編集に使うキー

編　集　キ　ー	機　　　　能
SHIFT	アルファベットの大文字，キートップの上段に書いてある特殊文字の入力に使用します．
CAPS	アルファベットを大文字で入力するときに使用します．
カナ	カタカナの入力に使用します．
GRPH	グラフィックシンボルの入力に使用します．
STOP	プログラムの中断や，行のキャンセルに使います．
TAB	カーソルを水平方向に８桁ずつ移動します．
CTRL	後述のコントロールコードの入力に使用します．
f・1 ～ f・5	キーの内容をユーザが定義して使用します．
INS DEL	SHIFT を押しながらインサートモードに入るとき，抜けるときに使用します． 単独で押すと DEL になり，カーソルの前にある文字が１文字削除されます．
↵	１行の入力に使用します．
↑ ↓ ← →	カーソルを上下左右に動かします．
HOME CLR	単独で押すと CLS になり，画面のクリアをします． SHIFT を押しながら押すと，HOME になり，カーソルを画面の左上(ホームポジション)に移動します．
ROLL UP　ROLL DOWN	後述のエディットモードにおいて，画面に表示されている行の前後を表示するときに使用します．
HELP	エラーによってプログラムの実行が中断されたとき，HELP を押すと，エラーのある行が表示されます．

2. ダイレクトモードによる実行

コマンドやステートメントをプログラムにしないで，単独で実行させるモードをダイレクトモードといいます．

ダイレクトモードでの実行は，右の例のように行います．N₈₈-BASICシステムディスクを使って，BASICをスタートさせてください．画面にはコマンド入力待ちのプロンプトが表示されています．

```
Ok
■
```

ここでは例として，ディスプレイ画面上に図形を描くLINE文を実行させてみましょう．右のようにキー入力してください．[CAPS]がロック(押された状態になっていること)されていれば大文字，そうでなければ小文字が入力されますが，どちらでもかまいません．

```
Ok
LINE -(300,100)■
```

[カナ]がロックされていると，カタカナが入力されてしまいますから，ロックを解除してから入力してください．

ここまで入力しただけでは，BASICは何も実行しません．コマンドを実行させるためには，[↵]を入力しなければなりません．[↵]を押すとどうなるでしょう．

```
Ok
LINE -(300,100)
Ok
■
```

LINE文が実行されて画面に直線が描かれました．

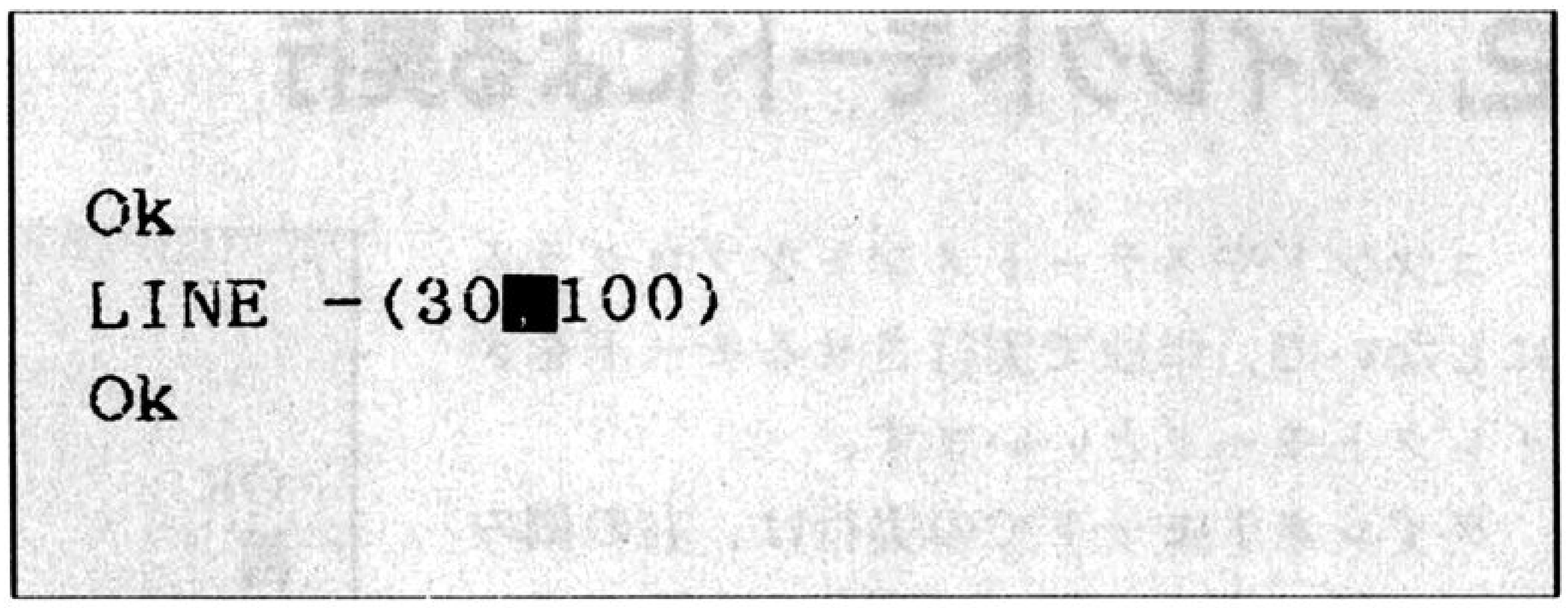
エラーが出てしまった場合は，もう一度正確に入力しなおしてみてください．画面に何も表示されない場合は，リセットボタンを押して，もう一度BASICの起動からやりなおしてください．

次にスクリーンエディタを使って，LINE −(300,100)をLINE −(30,200)になおして実行してみます．

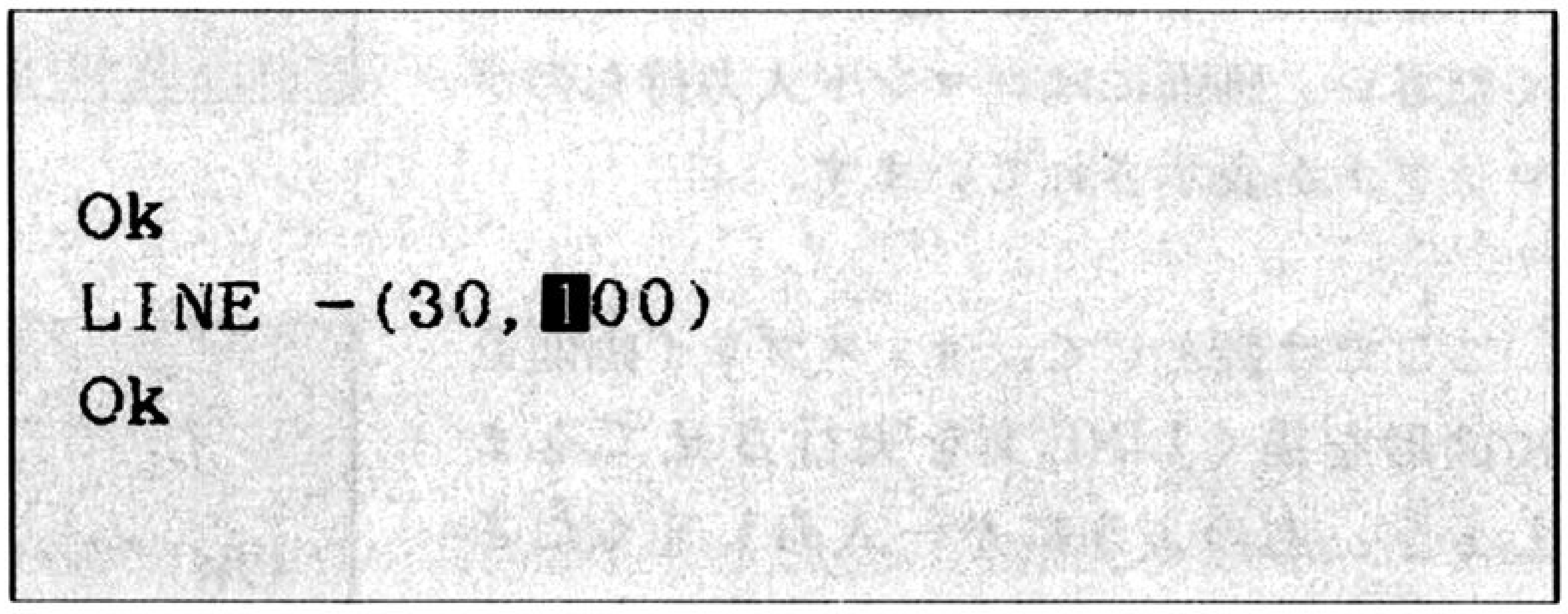
[↑]と[→]を使って，カーソルを右の図のように，,(カンマ)の位置まで移動させます．

```
Ok
LINE -(300■100)
Ok
```

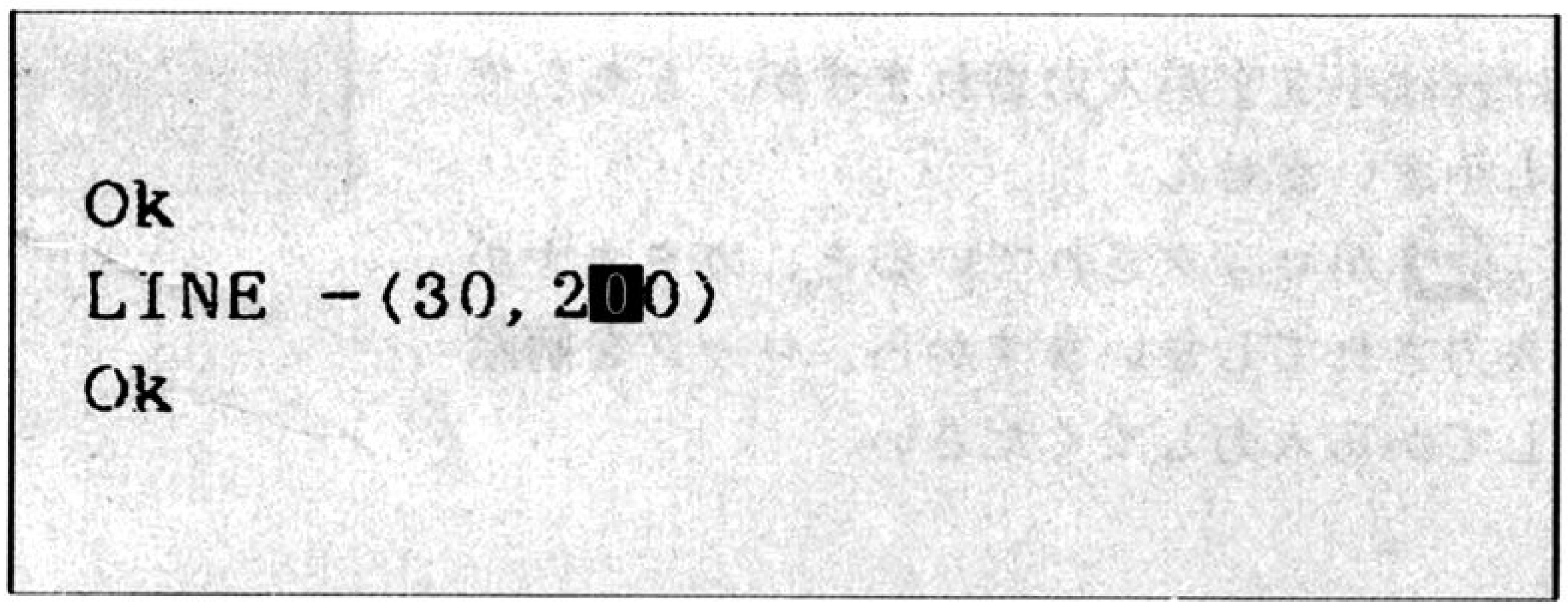
0を1つ削除するには，ここで[INS DEL]を押します．



次に，100を200に変えるためにカーソルを1の位置まで移動します．



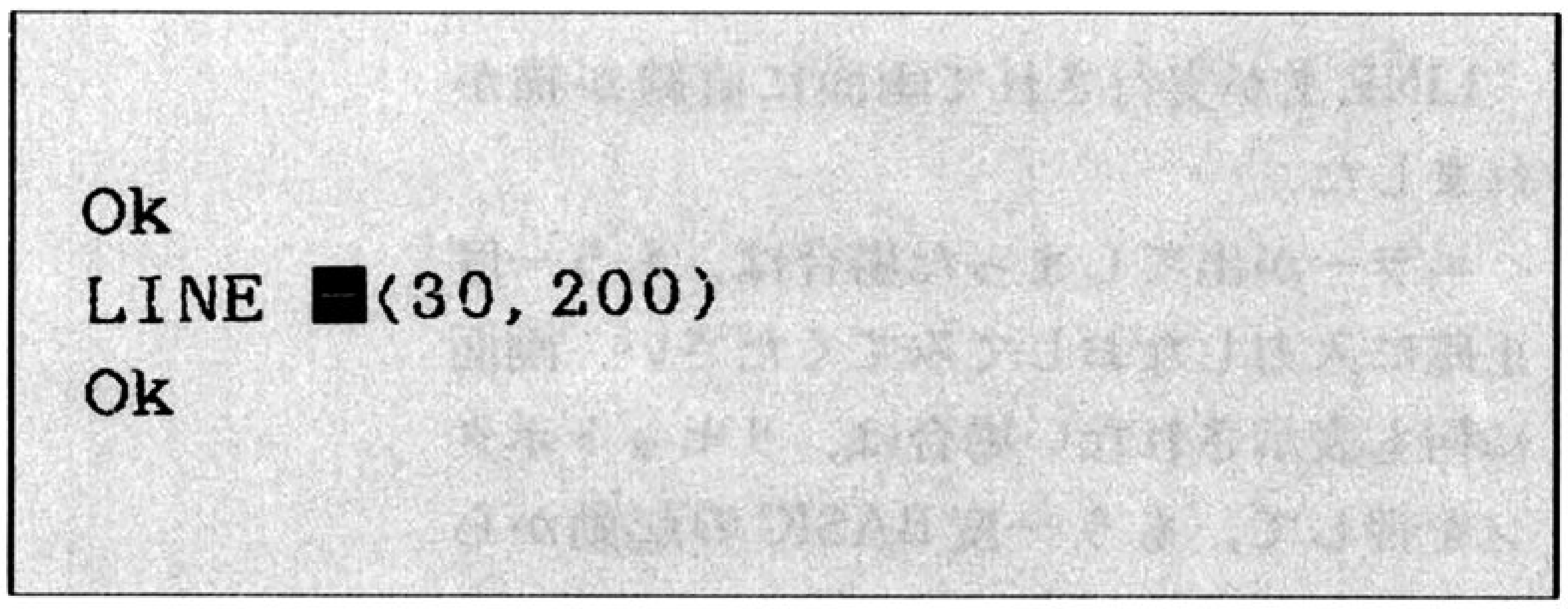
ここで2を入力すれば，100を200に変更できます．



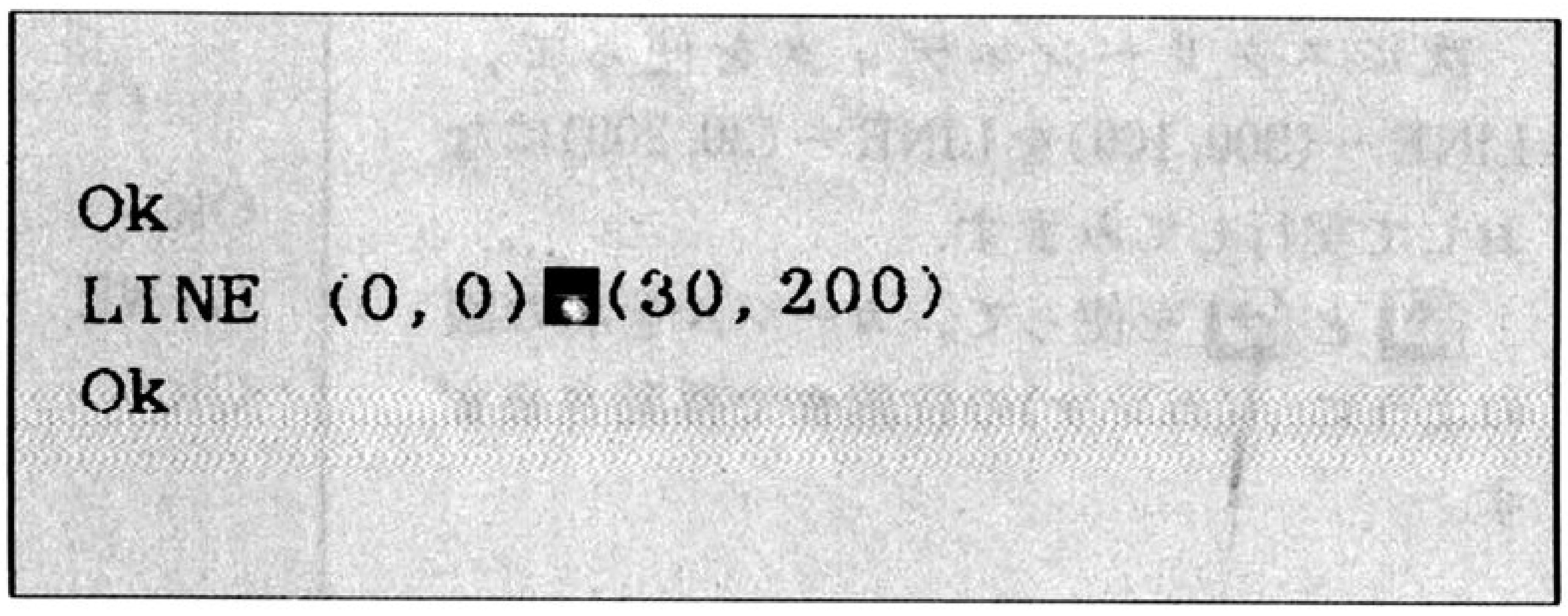
これで修正が終わりました．[↵]を押せば修正された命令が実行されます．

さて，次は挿入の練習をしてみましょう．先ほど実行した命令をLINE (0, 0)－(30, 200), 2, BFに変えます．

まず[↑]と[→]を使って，カーソルを右の図の位置まで移動させます．



ここで，[SHIFT]＋[INS DEL]（[SHIFT]を押しながら[INS DEL]を押します．）を押すとインサートモードになります．(0, 0)と入力すると，カーソルの位置から右側にある文字がひとつずつ右にずれていきます．

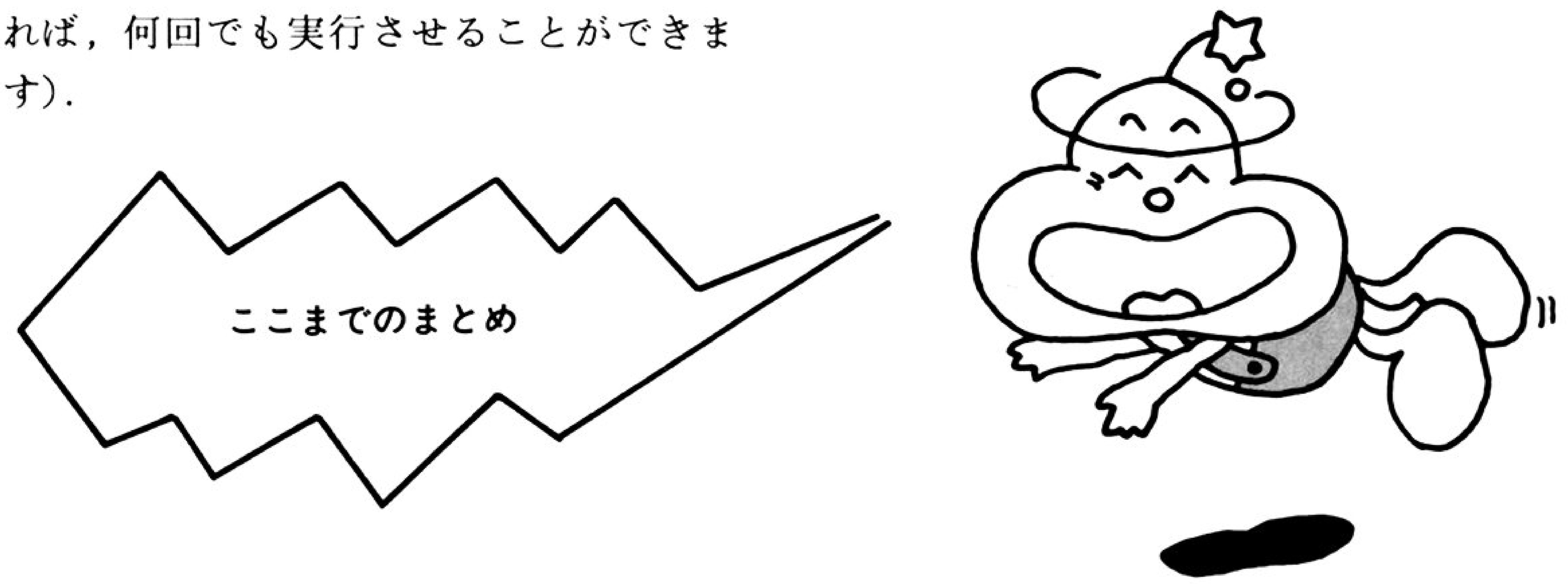


命令の最後に，2，BFを追加します．まずカーソルを右端の）の右に移動し，そこで入力してください．

```
Ok
LINE (0,0)-(30,200),2,BF█
Ok
█
```

修正が終わりましたので，[↵]を押せば実行されます．

このように，画面に表示されている文字は，修正して何度でも実行させることができます(修正しなくても，文の表示されている行にカーソルを移動して[↵]を入力すれば，何回でも実行させることができます)．



カーソルの直前の文字の消去

文字を入力しているときに入力した文字を消去するには，[INS DEL]を押します．すると，カーソルの前の1文字が消去されます．[CTRL]＋[H]でも同じ機能を実現できます．

途中の文字の修正

途中で押し間違えた文字を正しい文字に置き換えたい場合は，カーソル移動キー([↑][↓][←][→])を使用して，間違った文字のところへ移動させ，正しい文字に置き換えた後，再びカーソル移動キーで，入力を中断したところへ戻って入力を続けます．

途中の文字の消去

途中で押し間違えた文字を消去したい場合は，カーソル移動キー([↑][↓][←][→])を使用して，消去したい文字の次の文字のところへカーソルを移動させて，[INS DEL]を入力します．この場合，カーソルの前の文字が消去され，カーソルから右の

文字列が１文字分左へ移動します．

文字の挿入

何文字か途中で挿入したい場合は，文字を挿入したい位置の次の文字にカーソルを移動させ [SHIFT] ＋ [INS DEL] を入力します．これでインサートモードとなり，これから後で入力される文字をカーソルの前に挿入していきます．インサートモードから抜け出すには，再び [SHIFT] ＋ [INS DEL] を入力するか，カーソル移動キーかリターンキー（ [↵] ）を入力します（ [SHIFT] ＋ [INS DEL] は，[CTRL] ＋ [R] で代用できます）．

3. プログラムモードによる実行

BASICのコマンドやステートメントの集まりを続けて実行させるときは，プログラムを作り，RUN コマンドによって実行します．プログラムを入力するモードをプログラムモードといい，前述のダイレクトモードとは次の点が異なります．

1. 文の先頭に，1〜65529までの数字を付けて入力します．
2. [↵] で１行(255文字以内)を入力しますが，実行はされません．実行するには，RUN コマンドまたはGOTO 文を使います．
3. [↵] によって１行を入力しても，"Ok" プロンプトは表示されません．

ここでは，プログラムモードの入力の例として次のような短いプログラムを入力し，変更，追加，削除などの方法を具体的に説明します．



3.1 プログラムの新規作成

まずNEWコマンドを実行し，メモリ上にはプログラムが何もない状態にしておいてから次のプログラムを入力してみます．大文字でも小文字でもかまいません．

各行とも ↵ の入力によって，プログラムとしてメモリ上に格納されます（↵ マークは画面上には表示されません）．

```
10 FOR I=1 TO 10↵
20 NEXT I↵
30 END↵
█
```

プログラムを新しく入力したり，変更したりしたら，必ずLISTコマンドで正しく入力されたかどうかを確かめます．LIST ↵ と入力すると，右のように表示されます．

```
LIST↵
10 FOR I=1 TO 10
20 NEXT I
30 END
Ok
█
```

もしもLISTコマンドによって表示されたプログラムに誤りがあったら，ダイレクトモードの項で述べたようにして訂正します．このときも，↵ を入力しなければ何も変更したことになりません．プログラムを変更するときには，必ず ↵ を入力しなければなりません．

RUN ↵ と入力すると，右のプログラムが実行されます．

```
RUN↵
Ok
█
```

もし，エラーメッセージが表示されたら，LISTコマンドでプログラムを表示し，カーソルを動かしてエラーのあった部分を修正し，↵ を押してプログラムを変更すればいいのです．

スクリーンエディタの機能を用いてプログラムの訂正を行った場合は，各行番号の訂正が終わるたびに，必ずリターンキー（↵）を入力してください．

3.2　行の挿入と追加

BASICのプログラムは，行番号の小さいものから順に実行されます．したがって，行番号10と20の間に新しく行を挿入する場合は，11～19の行番号を付けた行を新しく入力すればよいのです．行番号30の後に追加する場合は，31以上の行番号を付けます．

行番号を15，16として，右のように入力してみます．

```
15 CIRCLE (300,100),I*20⏎
16 BEEP⏎
```

プログラムの変更を行いましたから，LISTコマンドを実行して確かめてみます．

```
LIST⏎
10 FOR I=1 TO 10
15 CIRCLE (300,100),I*20
16 BEEP
20 NEXT I
30 END
Ok
█
```

RUN ⏎ と入力して実行させてみましょう．もしエラーが出たら，カーソルを誤りのある部分に移動してなおし，⏎ を押します．

3.3 行番号の整理

行をいくつか挿入して行番号が混み合ってきたときなどには，RENUMコマンドを使って行番号を整理します．右のようにRENUMコマンドを実行し，LISTコマンドでプログラムを表示してみましょう．

```
RENUM 1000
Ok
LIST
1000 FOR I=1 TO 10
1010 CIRCLE (300,100),I*20
1020 BEEP
1030 NEXT I
1040 END
Ok
█
```

RENUMコマンドは，各行番号の間の増分や，プログラムのどこから行番号を付け替えるかの指定もできます．詳しくは，BASICリファレンスマニュアルをご覧ください．

3.4 行の複写

1つのプログラムの中に，内容が同一の行を何箇所かに作る場合は，行番号の部分だけを書き換えるだけで行のコピーができます．

このプログラムに右のような2行を追加したい場合は，行番号1000と1030のコピーを作ればよいわけです．

```
10 FOR I=1 TO 10
20 NEXT
```

カーソルを右図の位置に移動し，[INS DEL] を2回押して [↵] を押してください．

```
LIST↵
1000█FOR I=1 TO 10
1010 CIRCLE (300,100),I*20
1020 BEEP
1030 NEXT I
1040 END
Ok
```

同じように，1030を20と書き換えて [↵] を押します．

```
LIST
10 FOR I=1 TO 10
1010 CIRCLE (300,100),I*20
1020 BEEP
20 NEXT I
1040 END
Ok
```

画面上からは，行番号1000と1030はなくなってしまいましたが，メモリ上には残っています．LIST コマンドを実行してプログラムを表示させてみましょう．

```
LIST↵
10 FOR I=1 TO 10
20 NEXT I
1000 FOR I=1 TO 10
1010 CIRCLE (300,100),I*20
1020 BEEP
1030 NEXT I
1040 END
Ok
█
```

3.5 行の削除

1行ずつの削除は，行番号だけを入力して ↵ を押します．たとえば，右のように入力すれば，行番号1020が削除されます．

```
1020↵
```

プログラムのある範囲をまとめて削除する場合は，DELETE コマンドが便利です．たとえば，右のように行います．

```
DELETE 1000-1010↵
```

プログラム全体を削除するときは，NEW コマンドを使用します．

4. 便利なエディット機能

前の項まででで，スクリーンエディタの基本的な使い方について説明しました．ここでは，プログラム作りに慣れたあと，より能率的にスクリーンエディタを使いたいときに便利な機能を紹介します．

4.1 コントロールコード

カーソルの移動や文字の削除には ↑，↓，←，→，INS/DEL などのキーを用いてきましたが，他にも右のようなコントロールコードが使用できます．

カーソルの移動

CTRL + B	カーソルをひとつ前のワードに戻す．
CTRL + F	カーソルを次のワードに進める．
CTRL + K	カーソルをホームポジションに移動する．
CTRL + X	カーソルを行の最後に移す．

文字の削除

CTRL + E	カーソルのある位置からその行の最後までを削除する．
CTRL + D	カーソルのあるところから1ワード削除する．
CTRL + Z	カーソルのある以置以降の画面に表示されている範囲を消去する．
CTRL + U	カーソルのある行すべて消去する．

1行の内容を2行にわたるように書きなおしたい場合には，区切りたい箇所でカーソルを止め，SHIFT＋INS DEL でインサートモードに切り替え，CTRL＋J を押すと，カーソルから後のその行の最後までの文が，次のラインへ移動します．そして，移動したラインの先頭にカーソルが移動します．この場合も，モードはインサートのままなので，移動した行の先頭に行番号を付加し，↵ を押せば，これまでの1行の内容が2行にわたって書かれたことになります．

また，インサートモードになっていないときに CTRL＋J を入力すると，次の行の先頭へカーソルを移動(ラインフィード)させます．

資料4　コントロールコードもあわせてご覧ください．

4.2 ロールアップとロールダウン

プログラムが数十行にもわたる場合には，画面上にはその一部しか表示することができませんが，ロールアップ/ロールダウン機能を使うと編集はずっと楽にできます．

この機能は，巻き物を見るように，画面に表示されている行の前後を順に表示させ，修正する部分を探すために使います．これにより，訂正したい部分を短時間で探すことができます．

エディットモードに入る

ロールアップ/ロールダウン機能は，通常のプログラムモードから，さらにエディットモードに移って使用します．以下の2つの場合に，エディットモードに移ります．

(1) エラーによってプログラムが止まり，エラーの含まれる行が画面に表示されているとき(すべてのエラーが，このように表示されるわけではありません)．

(2) EDITコマンドを実行したとき．

たとえば，プログラムを実行して右のようになったとき，

行の分割とラインフィードの挿入

CTRL＋J

```
RUN ↵
Illegal function call in 100
Ok
█
```

EDIT .↵と入力すると，右のように表示されます．

```
EDIT .↵
```

```
100 CIRCLE(X, Y), R
```

ロールアップダウンの方法

(1)または(2)の状態で前の行が見たい場合は，[ROLL DOWN]を入力します．このキーを押し続けると，前の行が次々と表示されていき，画面に入りきらない行は画面から消えてしまいます．しかし，エディットモードに入っているときは，[ROLL UP]を入力することにより，画面から消えてしまった行を再度表示させることができます．スクリーンエディタの機能を利用して行を訂正したあと，↵を入力してください．

エディットモードから抜ける

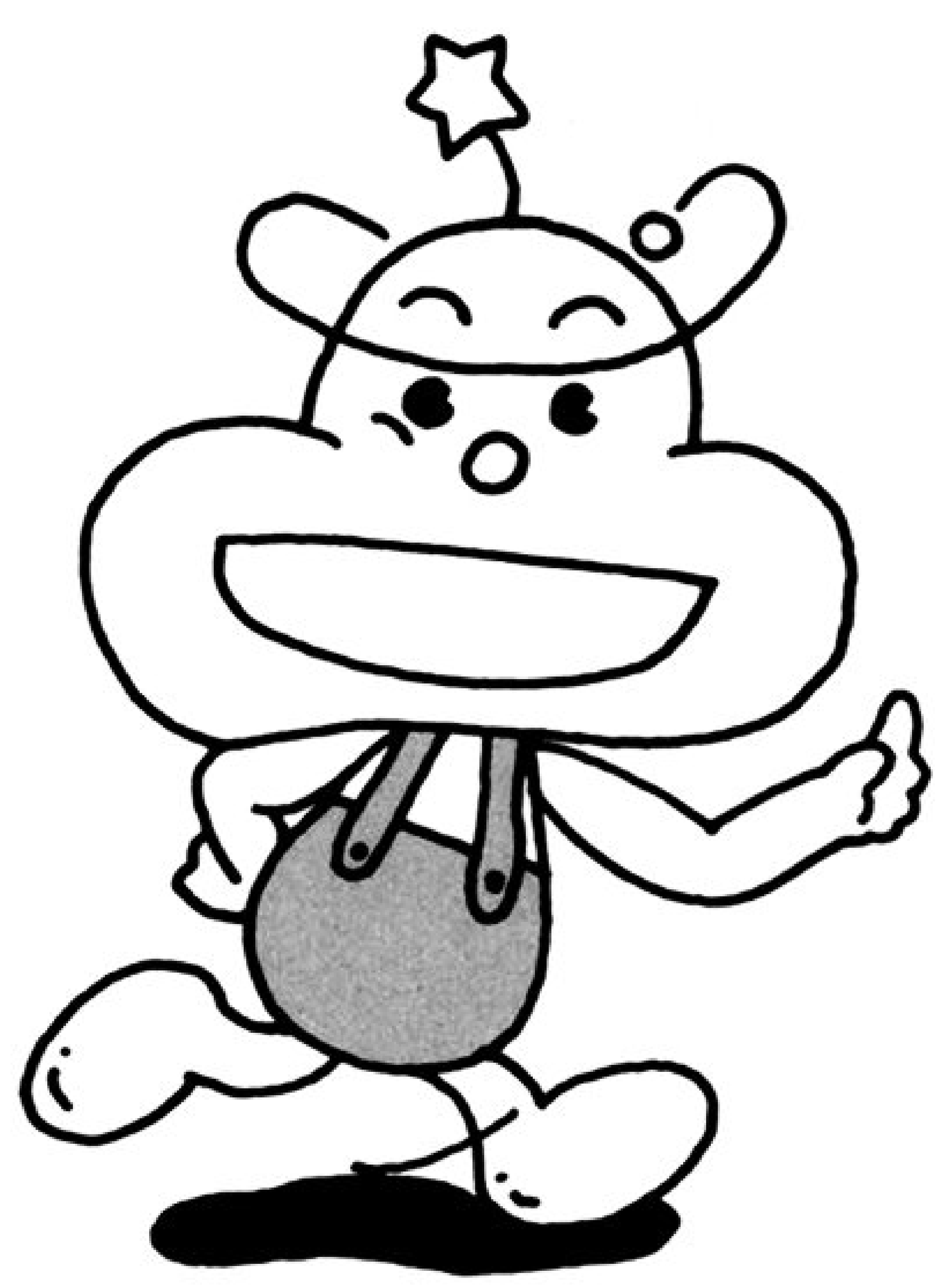
エディットモードから抜け出す場合は，[HOME CLR]を入力します．

4.3 HELPキーの使用

プログラムに何らかのエラーがある場合，その文で実行が中断されます．このときBASICは，エラーの種類と，そのエラーが起きた行番号をディスプレイに表示します．さらに，エラーの種類によってはその行を表示し，エラーの項目か，その次の項目の先頭にカーソルを表示します．

エラーのあった行を表示しない場合は，を入力します．すると，エラーメッセージで表示された行番号の文を表示して，エラーのある項目か，その次の項目にカーソルを表示します．

スクリーンエディタの機能を用いてエラーのあった行を訂正したあと，を入力します．



第2章 日本語文字の入力(N88-日本語BASIC)

N88-日本語BASICは，N88-BASICに日本語文字列を扱う機能を追加したBASICです．BASICリファレンスマニュアル 資料8の日本語コードにあげられている日本語文字を扱うことができます．これらの文字は，ちょうど日本語ワードプロセッサで文章を入力するように，キーボードから入力していきます．この章では，日本語文字の入力の方法を説明します．

1．日本語入力機能の概要

N88-日本語BASICの日本語入力機能の特長をまとめます．

(1) JIS第1水準および第2水準の文字が使用できます．

(2) カナ入力とローマ字入力のどちらも使えます．

(3) 熟語または文節単位で，カナを漢字混じりの語に変換することができます．

文法処理を行い，接頭語，接尾語を含んだ文節を一度に変換できます．また，アラビア数字を漢数字に変換することもできます．

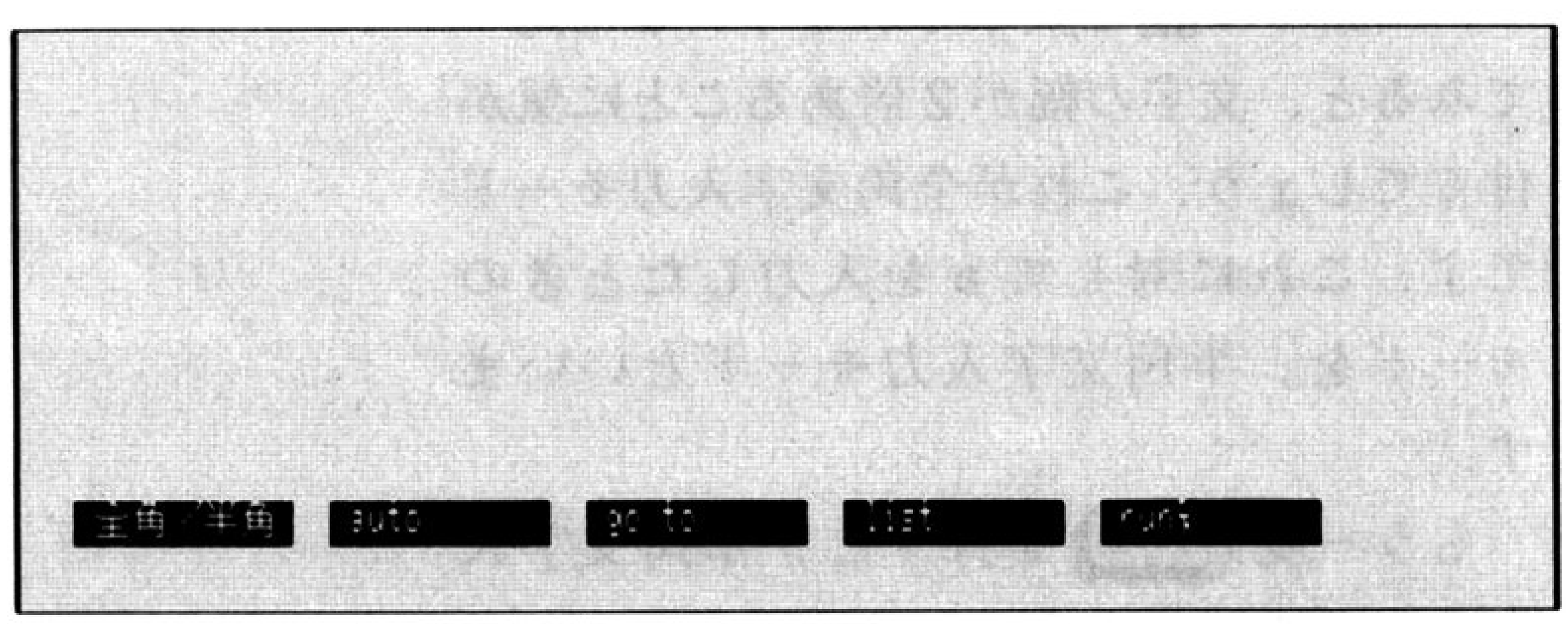
例） 9900→九千九百

(4) 学習機能のある，約4万語の辞書をシステムディスク上に持っていますから，使用頻度の高いものから順に，変換候補の順序が更新されていきます.* 使用していくに従って，ユーザに使いやすい辞書になっていくわけです．

* N88-日本語BASICシステムディスクにライトプロテクトシールが貼ってあるときは，辞書の学習機能は働きません．辞書を更新しながら使いたいときは，ライトプロテクトシールをはがしておいてください．

2．日本語モードの画面表示

N88-日本語BASICをスタートします.** 画面のいちばん下には，右のようにファンクションキーが表示されています．



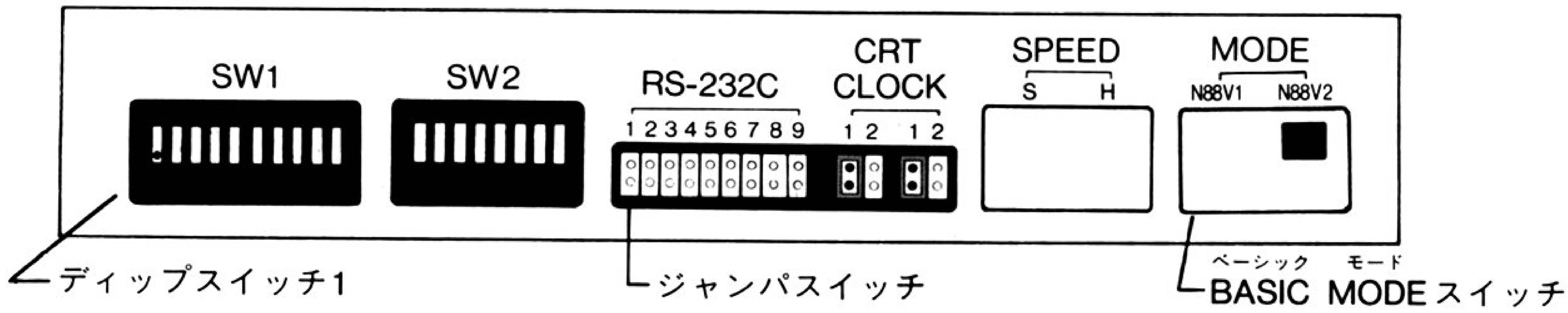
これは80文字/行のときの表示例です．

40文字/行のときは，f･1は 全／半 となっています．

** 次ページのN88-日本語BASICのスタートをご覧ください．

N88-日本語BASICのスタート

N88-日本語BASICは，ミニフロッピーディスクドライブが必要です．N88-日本語BASICシステムディスクを，本体の右側のドライブにセットし，スイッチ類は図のようにセットして電源を入れます．



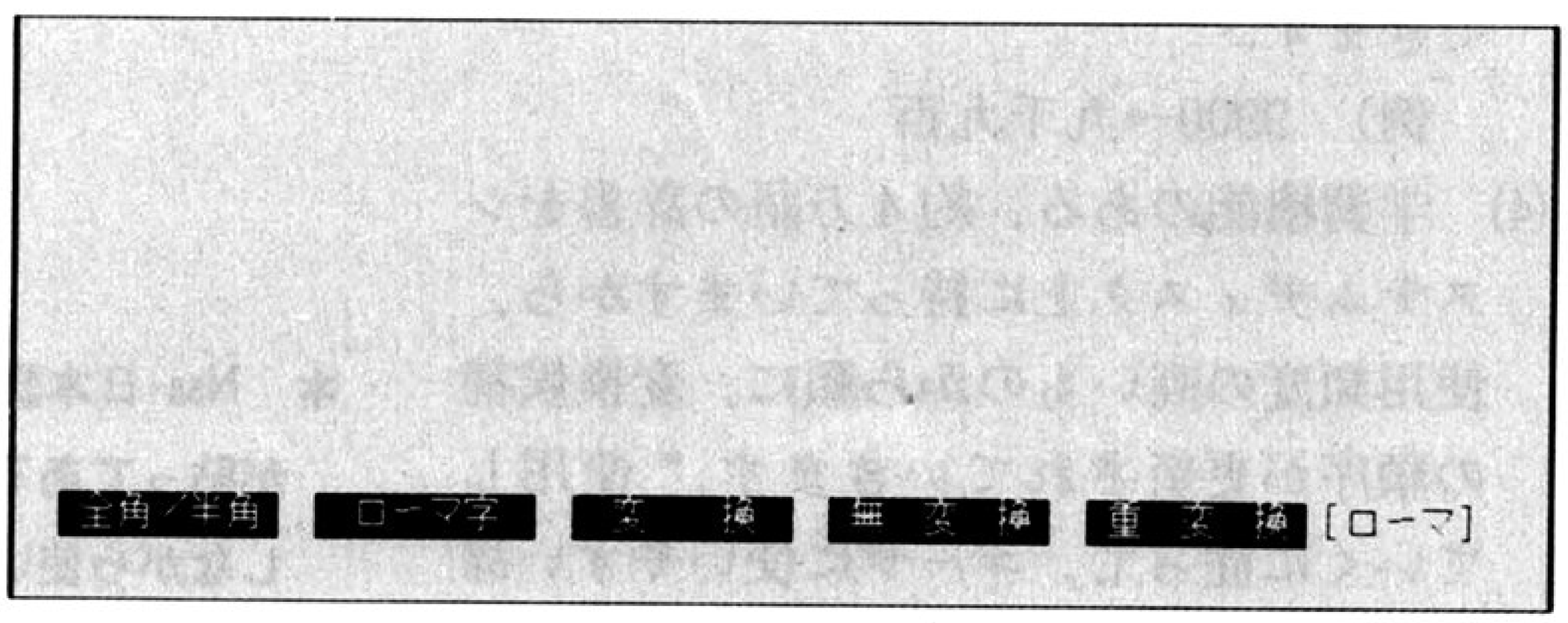
図に指定のないスイッチは，ご使用中のシステムに合わせて設定してください．

N88-日本語BASICのシステムディスクには，辞書や外字データが入っています．これらは，N88-日本語BASICの動作中に頻繁に読み出されますから，常にシステムディスクをセットしておかなければなりません．

[f・1]には，[全角／半角]が設定されています．この状態は"日本語モード"と呼ばれ，全角文字(日本語の文字など)と半角文字(英数字など)の両方を表示/入力できます(詳しくは，「第3章　日本語処理」をご覧ください)．

ここで([CAPS]と[カナ]が押されていない状態で)[A]を押してみると，画面には**a**が表示されます．

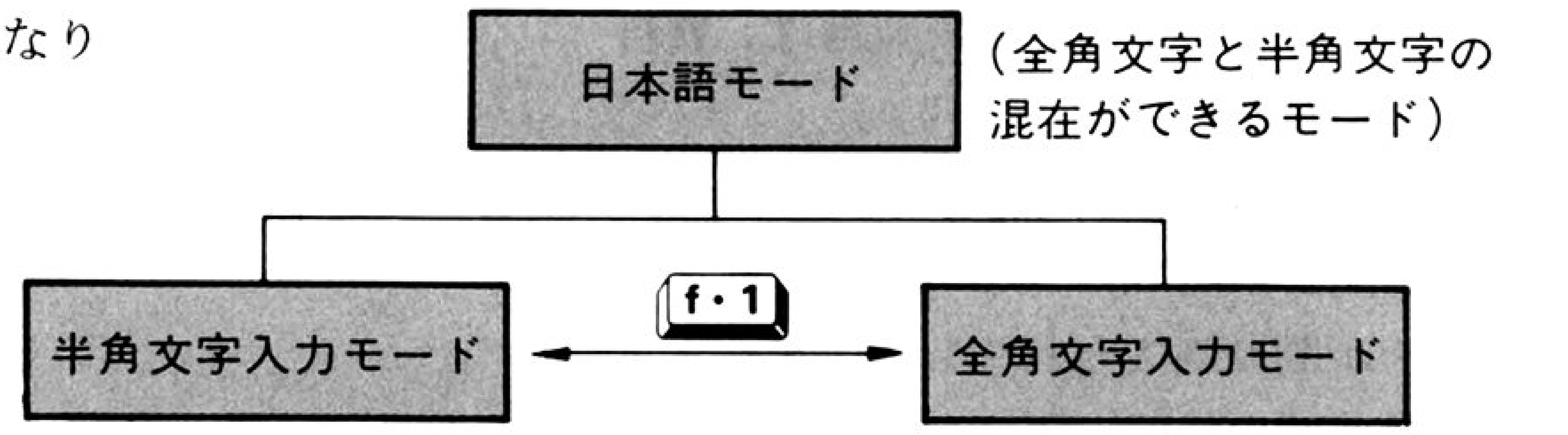
[f・1]([全角／半角])を押してみましょう．ファンクションキーは，右のように変わります．

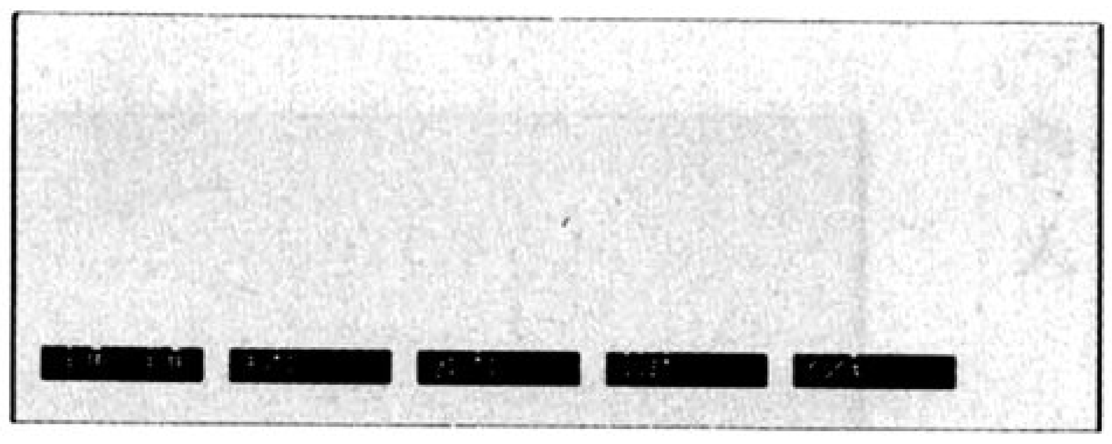


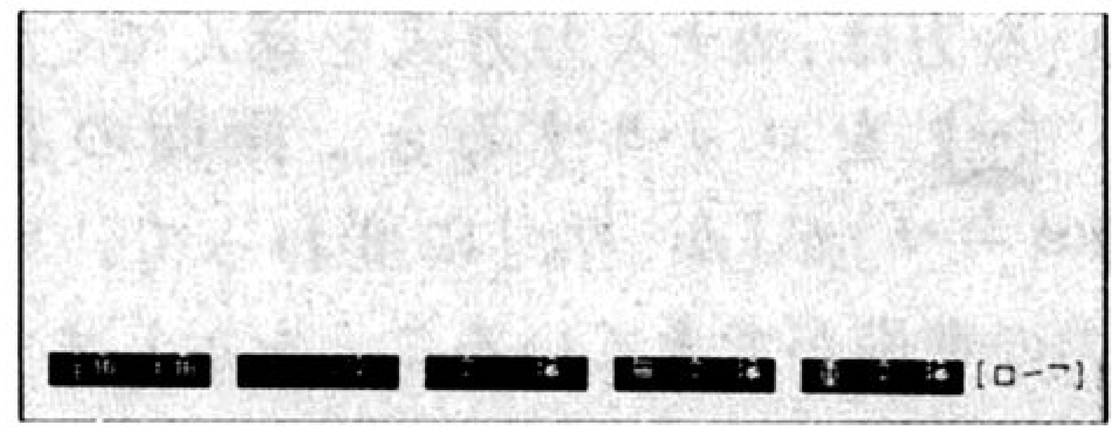
先ほどと同じように，[A]を押してみます．画面には**あ**と表示されます．**a**と比べてみると，文字の幅が2倍あることに気が付くでしょう．これが全角文字入力モードです．これに対して**a**を入力したときのモードを，半角文字入力モードといいます．

もう一度[f・1]を押すと，半角文字入力モードに戻ります．

これをまとめると，右の図のようになります．







[f・1] 以外のファンクションキーはユーザが定義できます．

すべてのファンクションキーは，全角文字の入力に使われているため，固定されています．

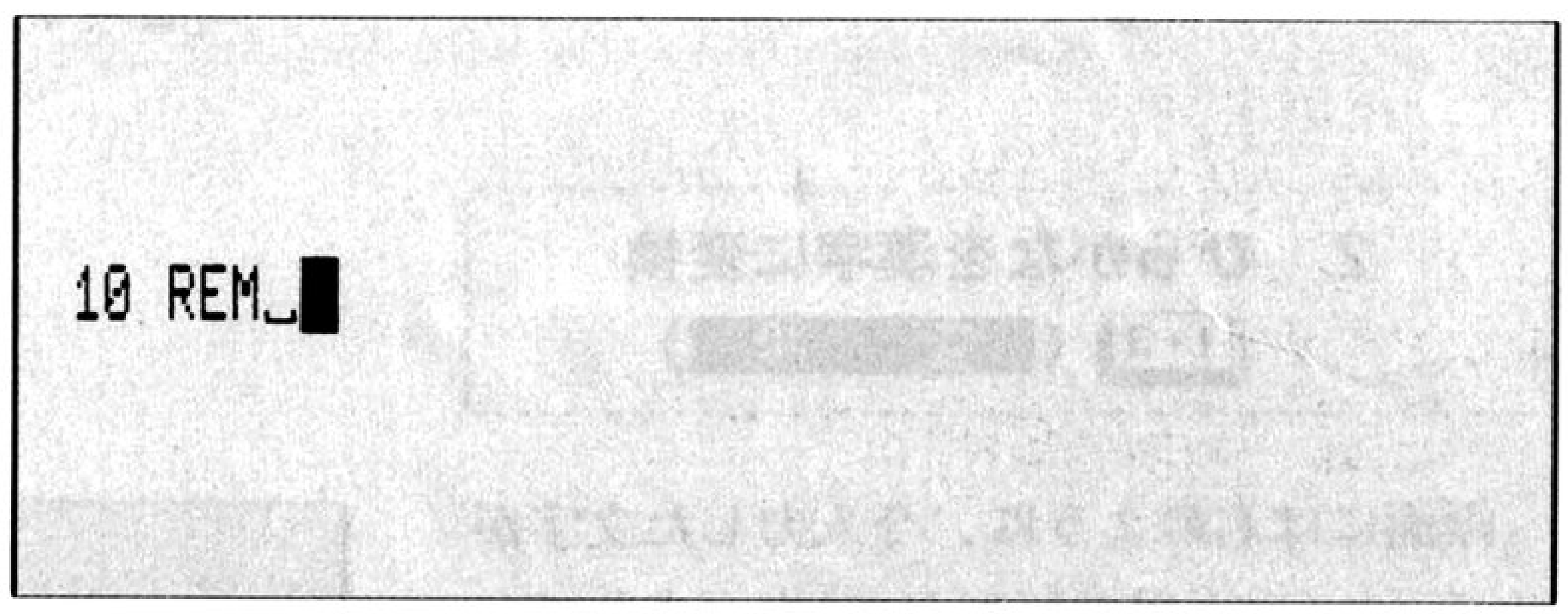
3. 入力の手順

実際に，右のようなプログラムを入力しながら，日本語の入力の手順を説明していきます．

```
10 REM 入力の練習
20 PRINT "カタカナとひらがなの入力"
30 DATA ａｂｃＡＢＣ１２３
40 A$="愛":B$="★":C$="凰"
```

まず半角文字入力モードで，右のように入力します．入力の方法がわからないときは，もう一度第1章を見なおしてください．

[f・1] を押して，全角文字入力モードに切り替えます．



␣はスペースを表します．

英字は，大文字でも小文字でもかまいません．

1. ローマ字入力とカナ入力 [カナ]

次に，"にゅうりょくの"とキーボードから入力して，"入力の"に変換します．

ひらがなを入力するには，ローマ字入力方式とカナ入力方式の2通りがあります．

ローマ字入力

英文のタイプライタに慣れている方なら，ローマ字で右のようにタイプすればよいでしょう．



↓

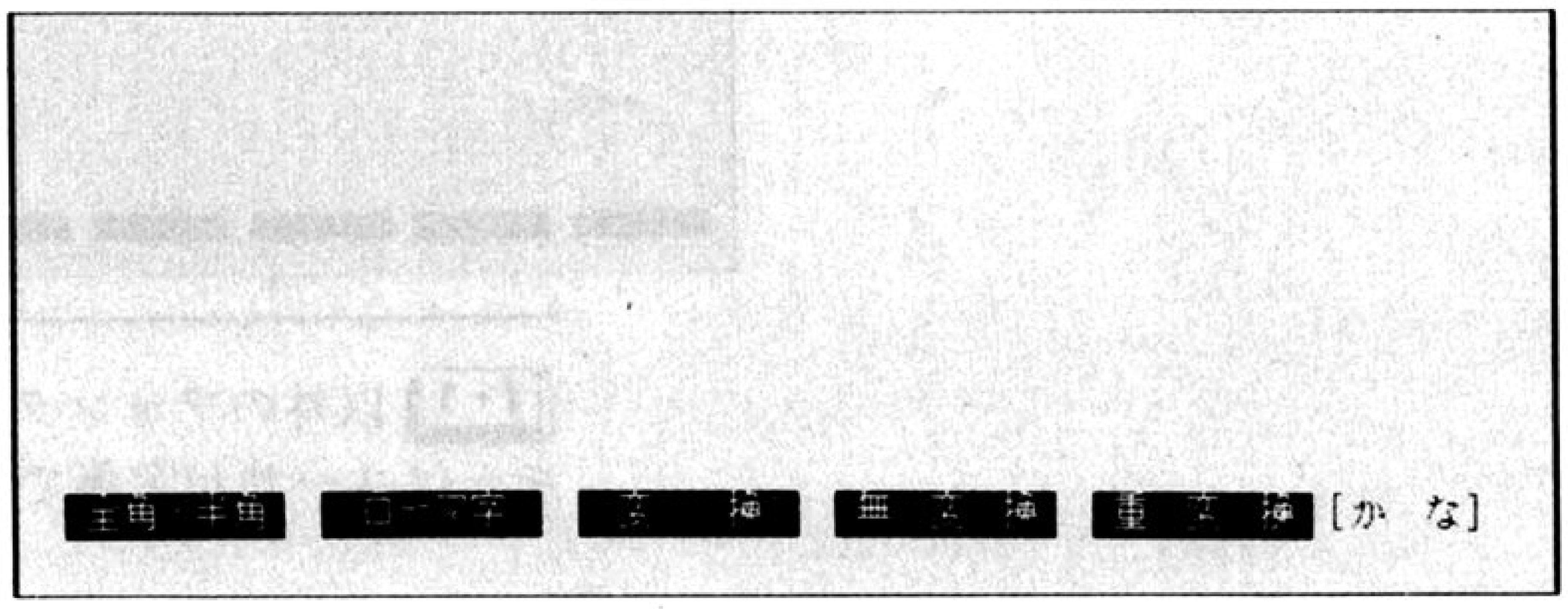
にゅうりょくの

ローマ字とひらがなの変換規則は，資料7をご覧ください．

カナ入力

PC-8801mkⅡMRのキーには，それぞれカタカナが割り当てられていて，キートップに記されています．カナの入力に慣れている方は，カナ入力方式を選んでください．

カナ をロックすると，画面の右下の［ローマ］が［か　な］に変わって，カナ入力の準備ができていることを示します．



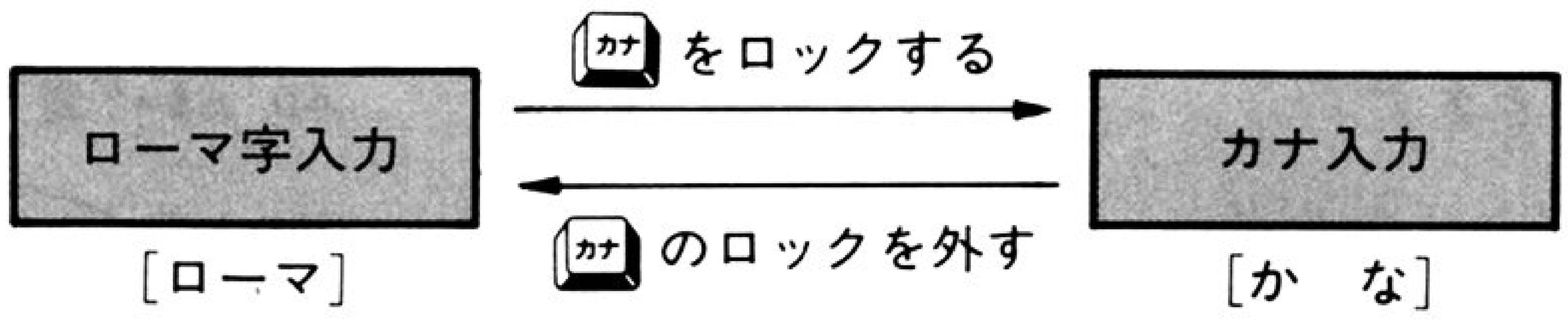
カナ がロックされた状態で，右のようにタイプします．



↓

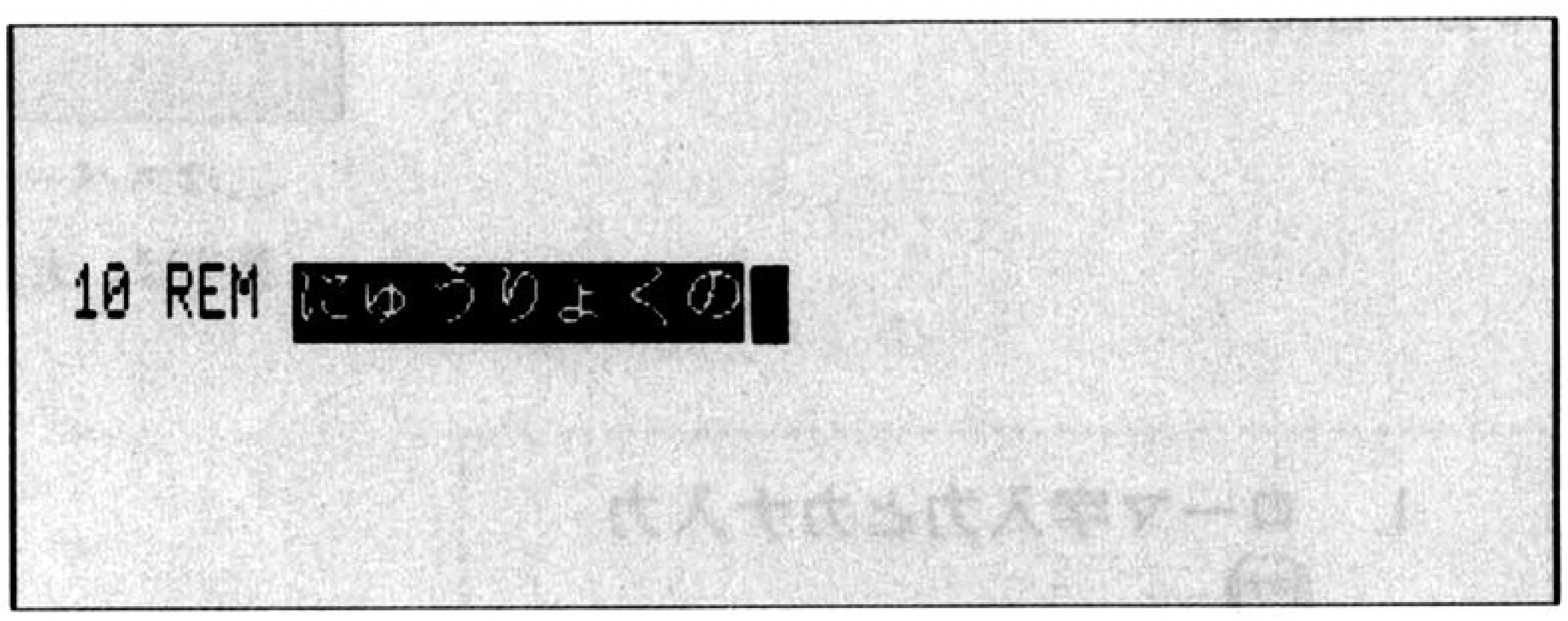
にゅうりょくの

ローマ字入力とカナ入力は，カナ で切り替えます．



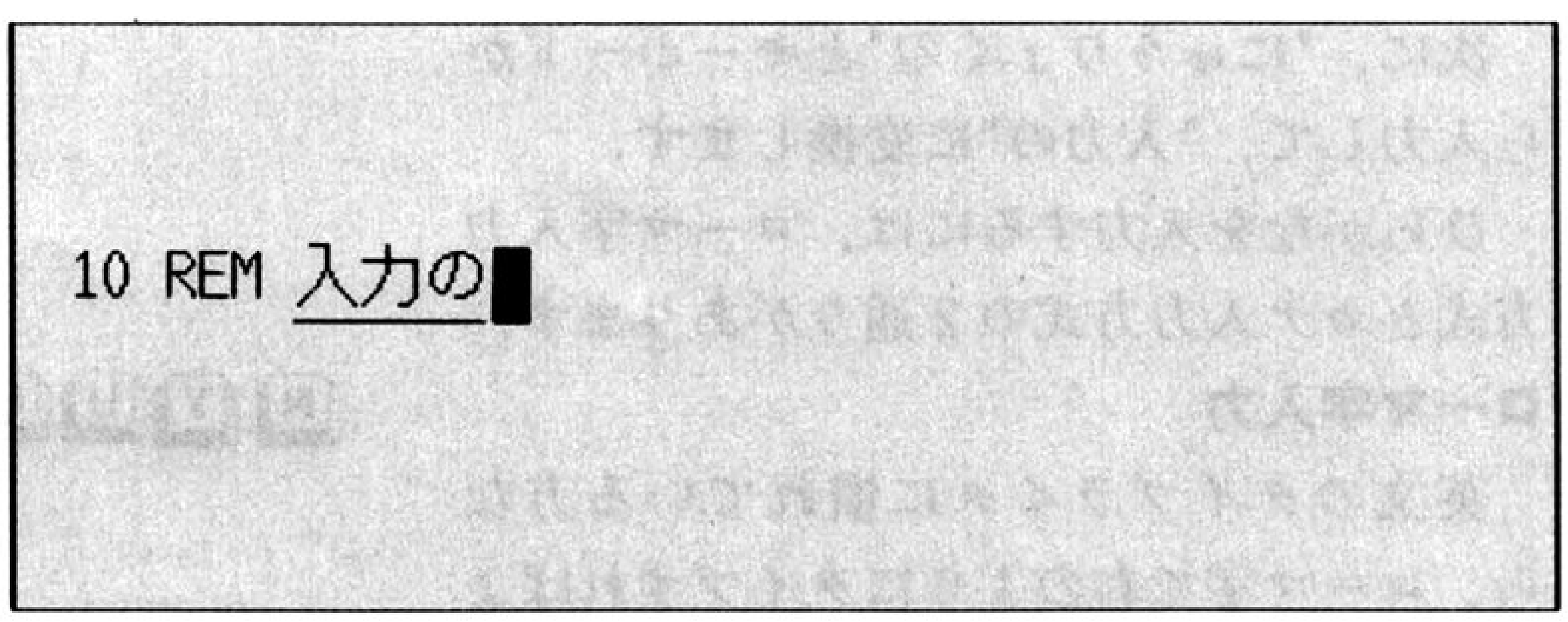
2. ひらがなを漢字に変換 f·3 (変　換)

画面には右のように，今入力した文字がリバース(白と黒が反転して)表示されています．



リバースになっている部分は，入力の途中であり，これから変換される部分であることを示します．

ここで f·3 *(変　換)を押すと，ひらがなは漢字に変換されます．



* スペース も f·3 と同様，変換に使用できます．

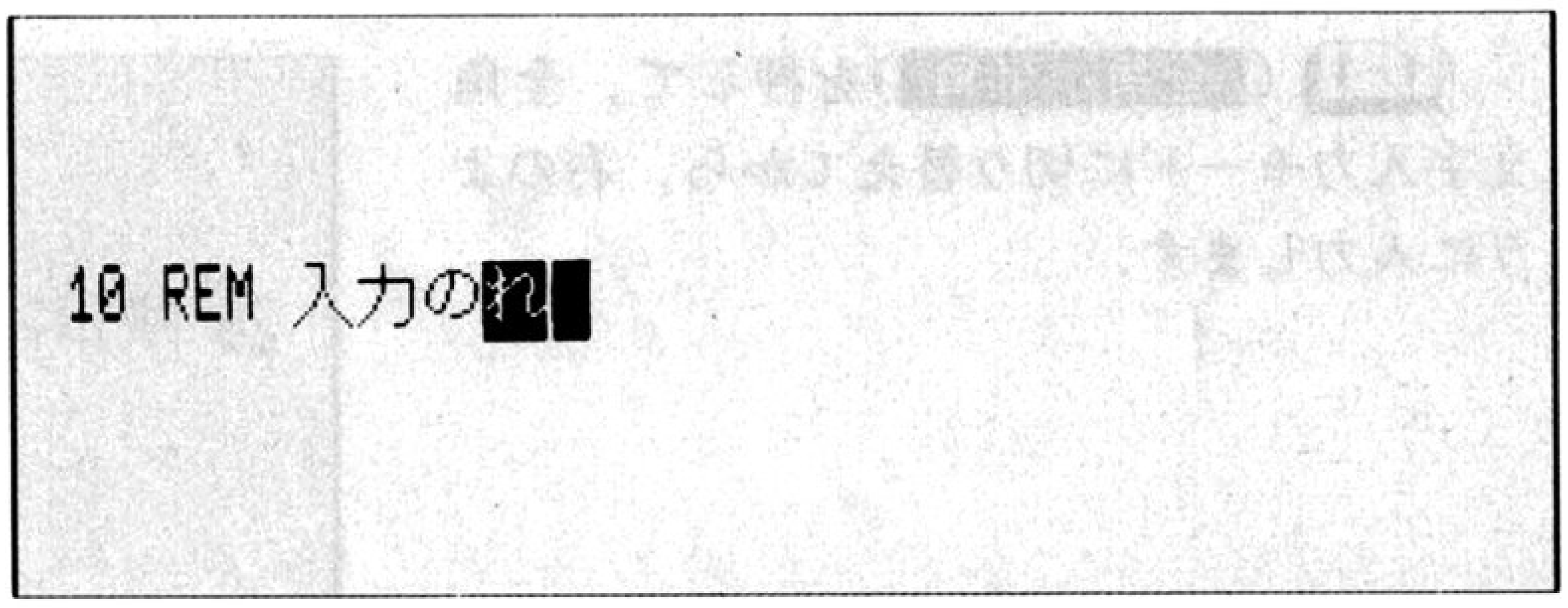
希望の漢字に変換できました．このとき，漢字に変換された"入力の"の部分はアンダーラインが付いています．

"にゅうりょくの"に対応する漢字まじりの語は"入力の"しかないので，ここでは [f･3] を一度だけ押すことによって，希望の語に変換されます．

同じ読みを持つ語がいくつかあるときは [f･3] を一度押しただけでは希望の語に変換されない場合があります．もし，希望の漢字に変換されなかった場合は，もう一度 [f･3] を押します．これによって，同じ読みを持つ別の漢字候補が現れます．

このように，[f･3] を押しながら漢字候補を選んでいきます．

次の"練習"を入力するためにローマ字入力の場合では [R]，カナ入力の場合は [レ] を押すと"入力の"の変換が終わった(決定された)とみなされて，ノーマル表示になります(アンダーラインやリバース表示はなくなります)．

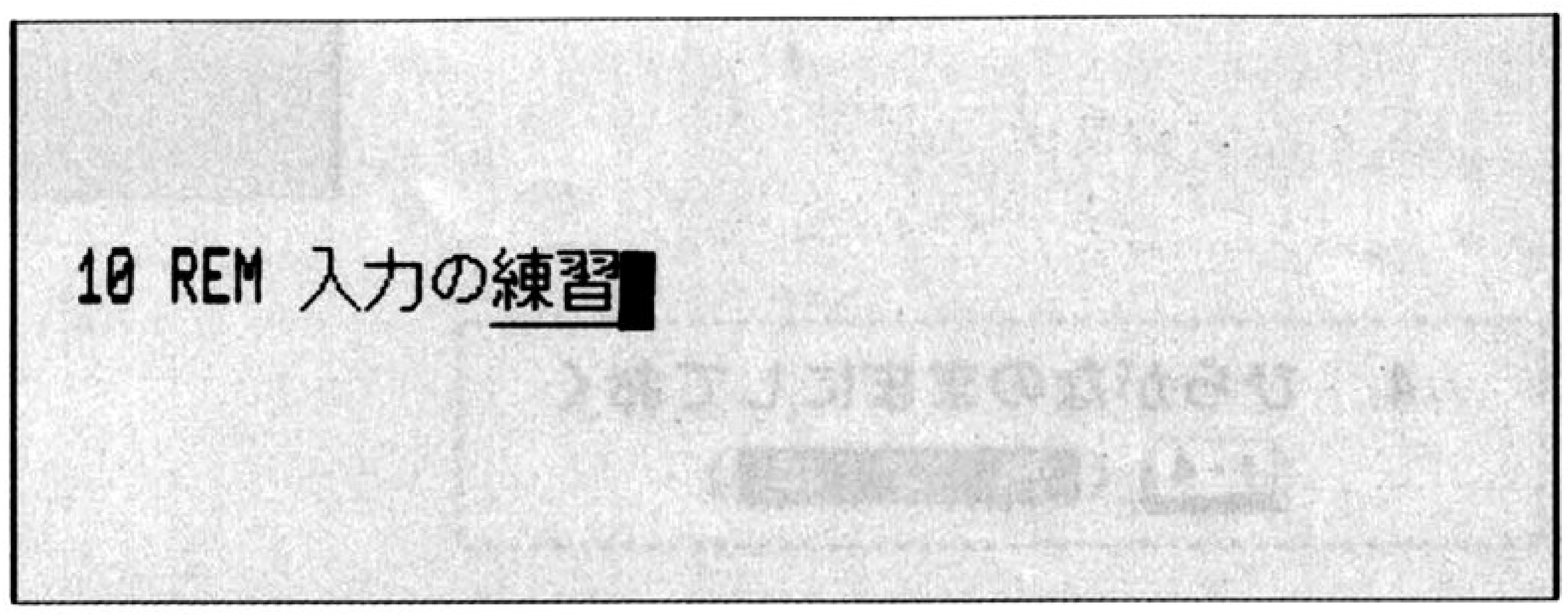


同じようにして，"練習"を入力してみましょう．

ローマ字入力では(1)のように入力し，カナ入力では(2)のようにキー入力します．

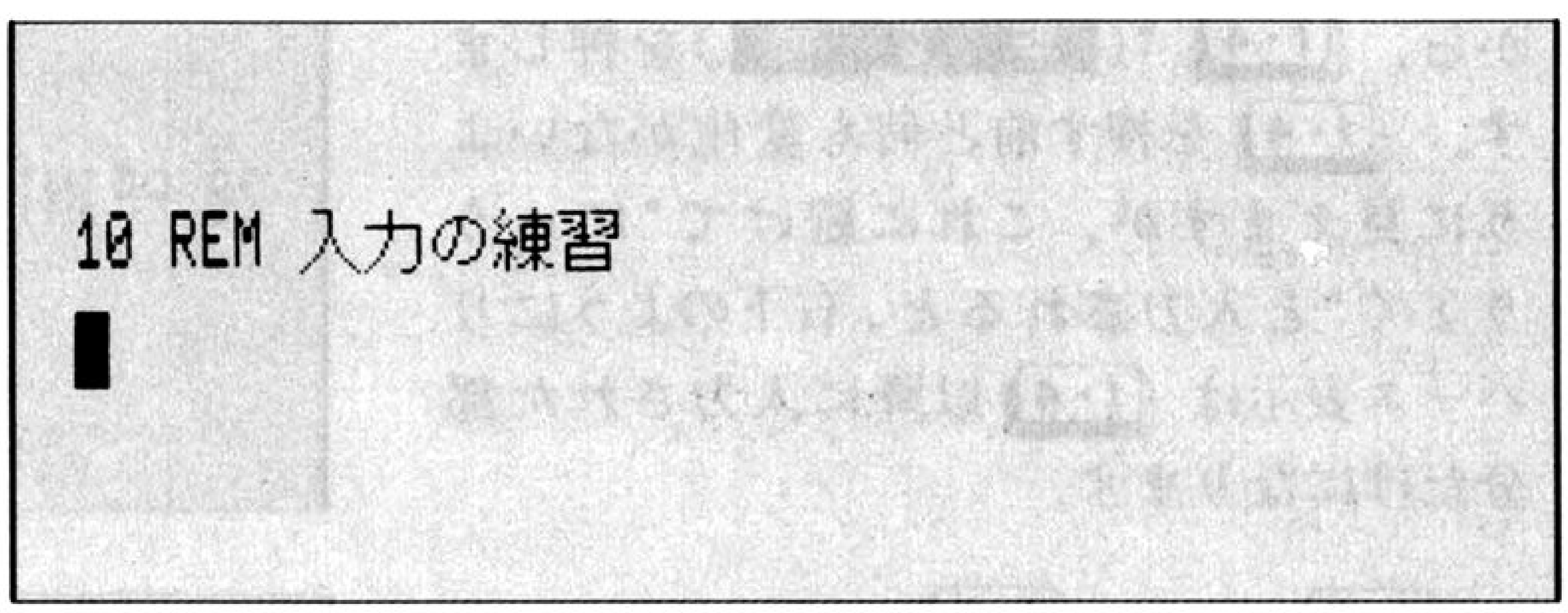
[R][E][N][S][H][U][U][f･3] ——(1)

[レ][ン][シ][SHIFT]+[ユ][ウ][f･3] ——(2)



行末で [↵] を押すと，"練習"が決定されます．

もう一度 [↵] を押せば，10行の入力が終了します．



[f・1]（全角／半角）を押して半角文字入力モードに切り替え，次のように入力してください．

```
20 PRINT␣"
```

3. ひらがなをカタカナに変換する [f・10]（カタカナ）

[f・1]（全角／半角）を押して，全角文字入力モードに切り替えてから，右のように入力します．

```
20 PRINT "かたかな
```

[f・10]（カタカナ）を押すと，カタカナに変換できます．

```
20 PRINT "カタカナ
```

4. ひらがなのままにしておく [f・4]（無　変　換）

続いて，"とひらがなの"と入力してから，[f・4] *（無　変　換）を押します．[f・4]を押す前と何も変化がないように見えますが，これに続けて"にゅうりょく"と入力されると，右下のようにリバース表示は[f・4]以降に入力された部分だけになります．

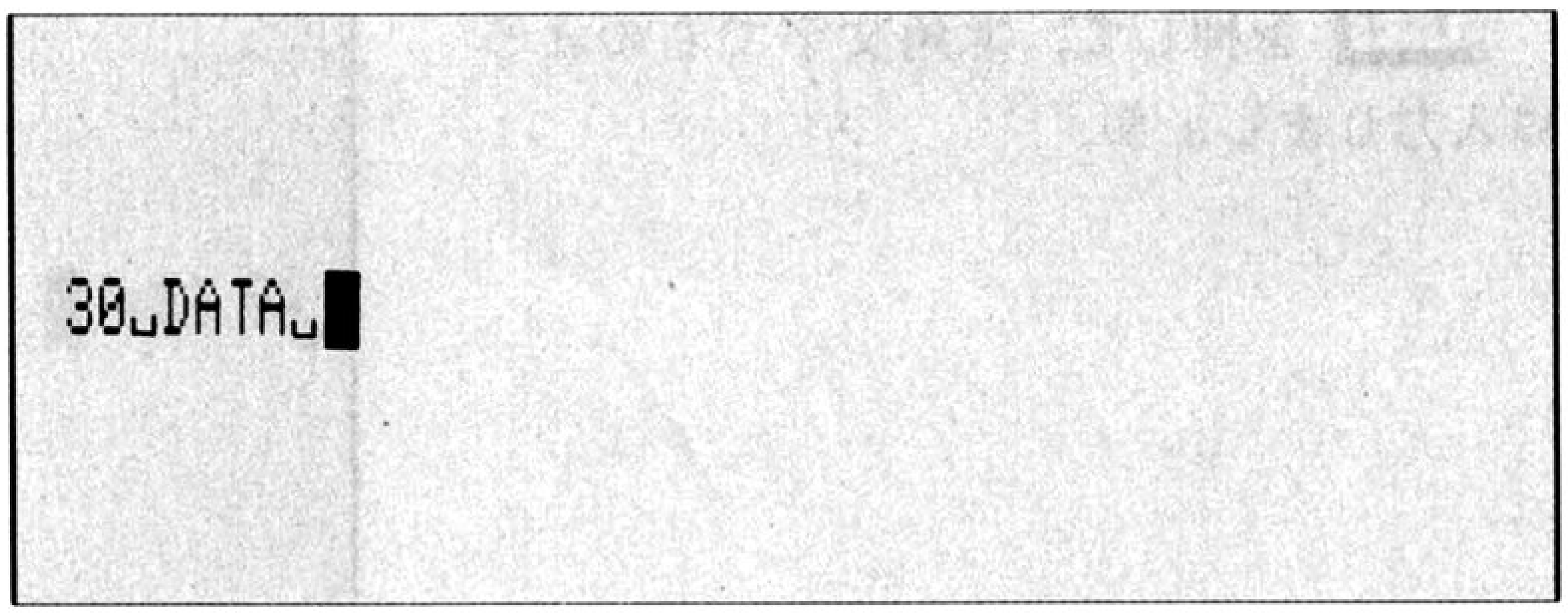
```
20 PRINT "カタカナとひらがなの
```

[f・3]そして[f・1]と押し，行末の"を入力します．[↵]を押せば，20行は完成です．

```
20 PRINT "カタカナとひらがなのにゅうりょく
```

* [SHIFT] + [スペース] も [f・4] と同様，無変換に使用できます．

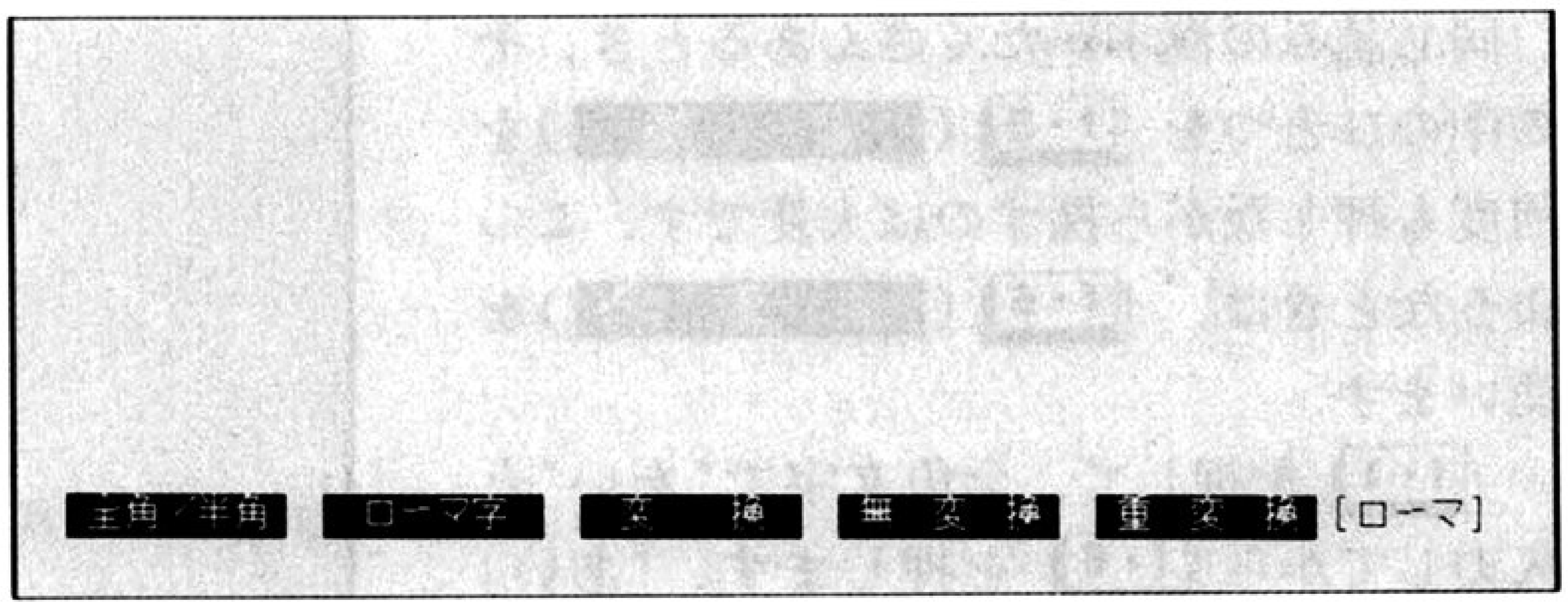
30行目は，まず半角文字で右のように入力します．



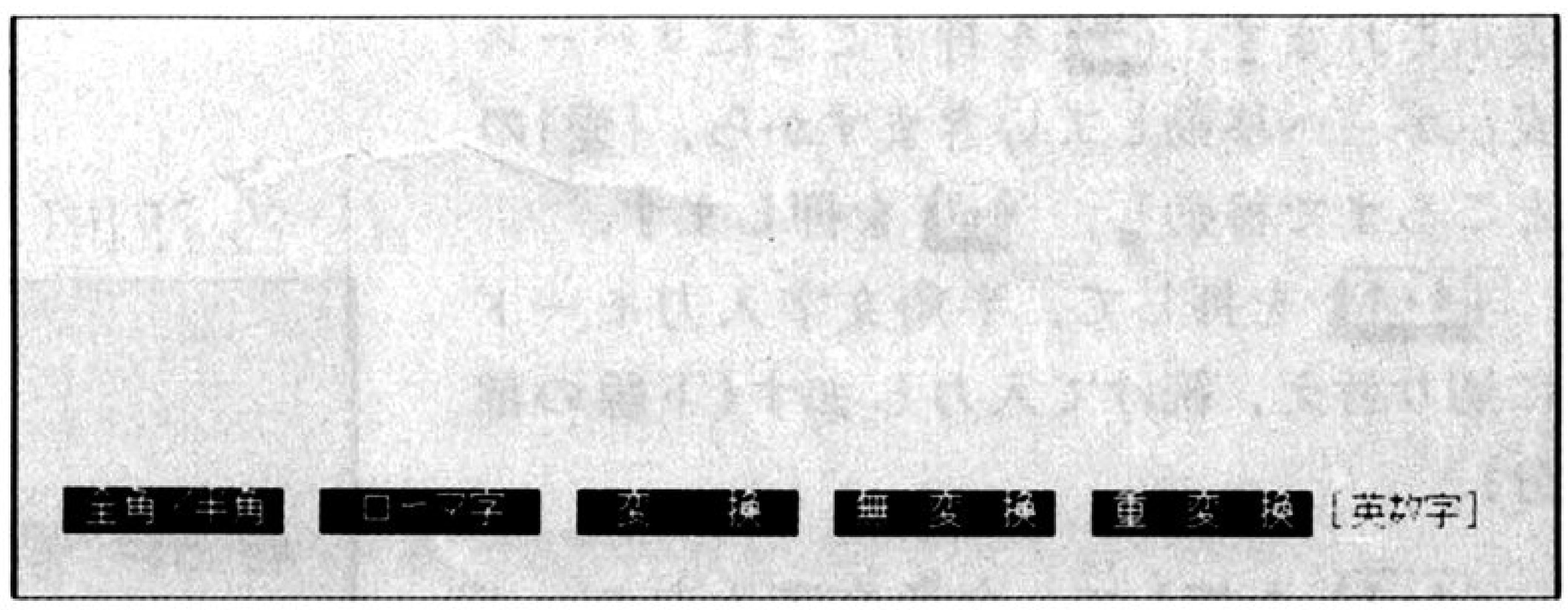
[f・1] を押して，全角文字入力モード にします．

5. 英数字を全角で入力 [f・2]（ローマ字）

（まず [カナ] がロックされていたら，ロックを外してください．）画面の右下端には［ローマ］と表示されていますね．



ここで [f・2]（ローマ字）を押すと，右下の［ローマ］が［英数字］に変わります．



[f・2] を何度か押してみると，［ローマ］と［英数字］が，交互に切り替わります．

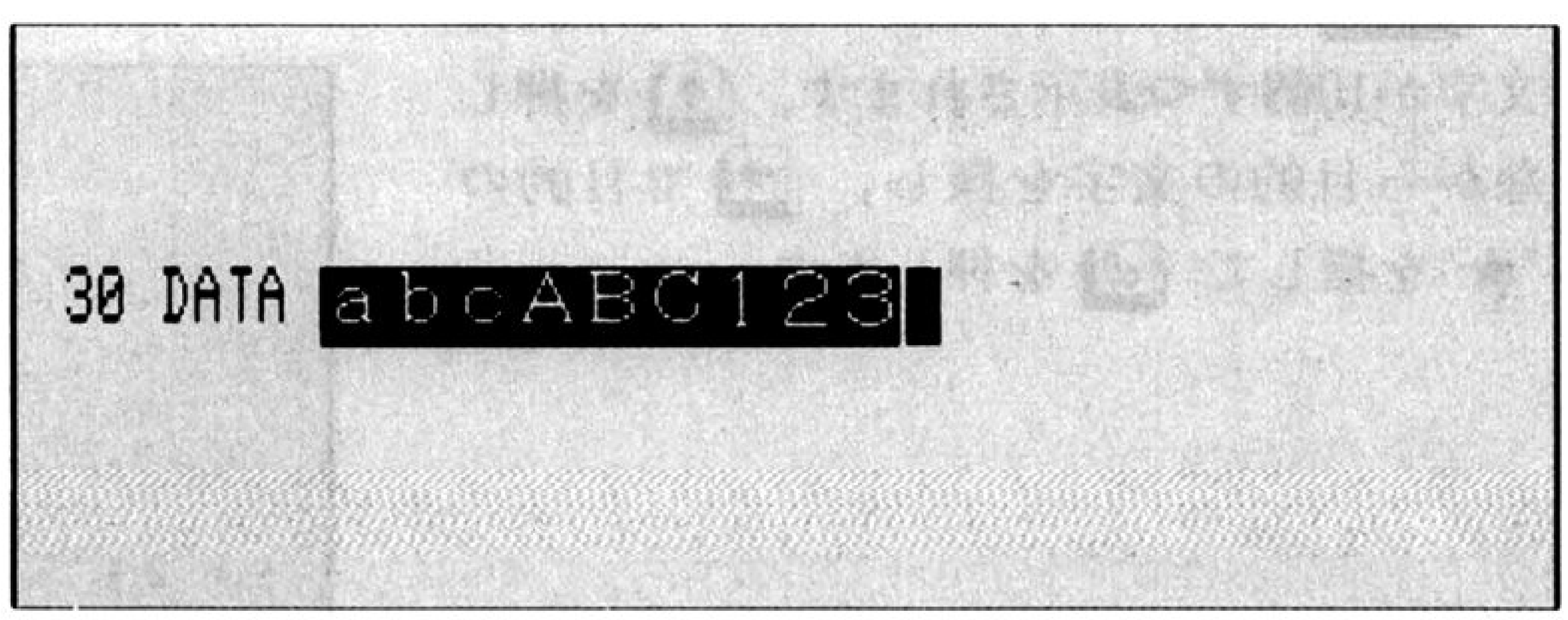
［ローマ］ ←[f・2]→ **［英数字］**

つまり [f・2]（ローマ字）は，英字をそのまま全角文字で入力するか（［英数字］），ローマ字入力としてひらがなに変換しながら入力するか（［ローマ］）の切り替えをします．

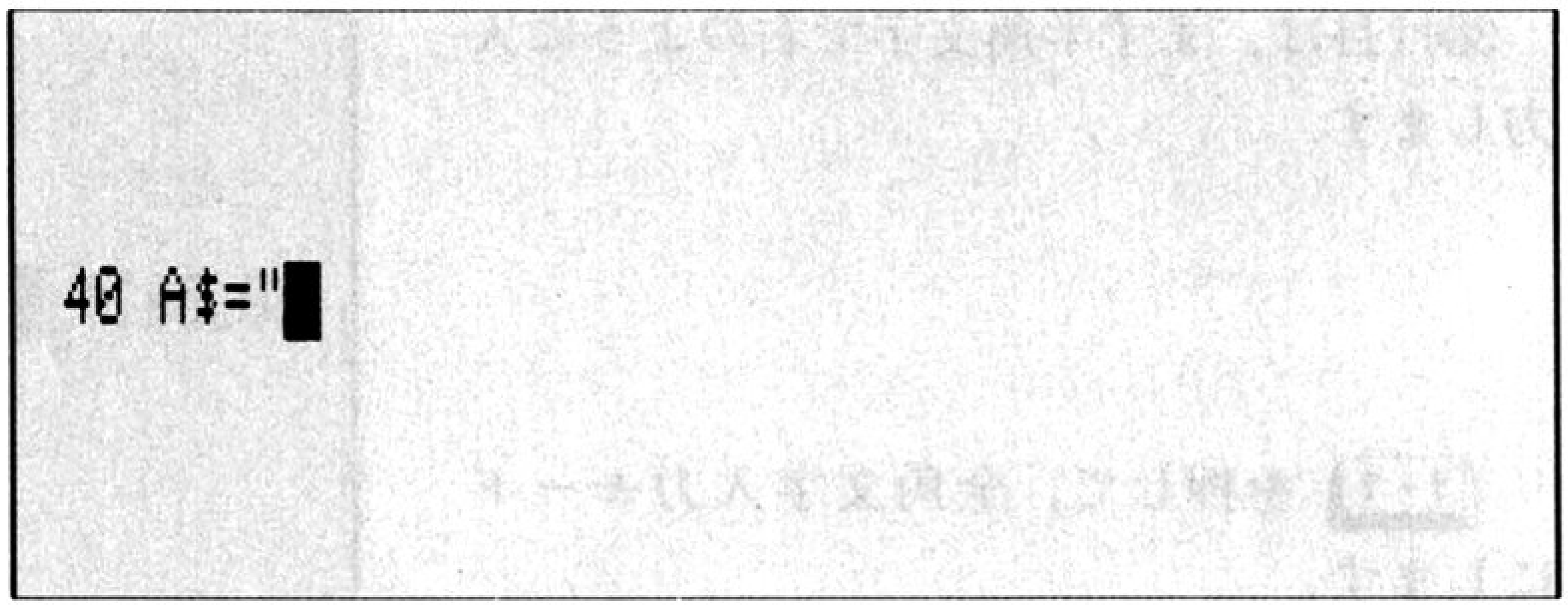
ただし，[f・2] がこのような働きをするのは，[カナ] がロックされていない場合に限ります．

［英数字］の状態でキー入力を行うと，全角の英数字の入力ができます．半角文字を入力するのと同じ領域です．

右のように入力してから [↵] を2回押せば，30行も完成です．



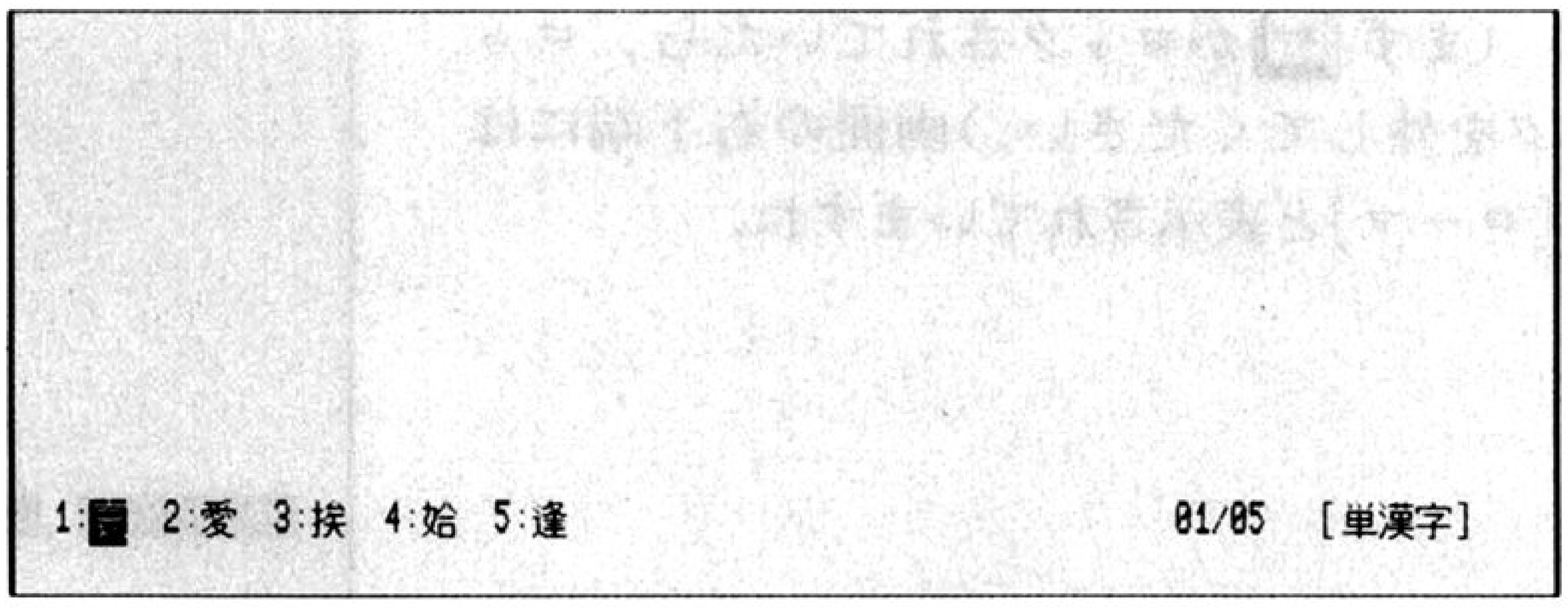
f·1 を押して，半角文字で右のように入力しましょう．



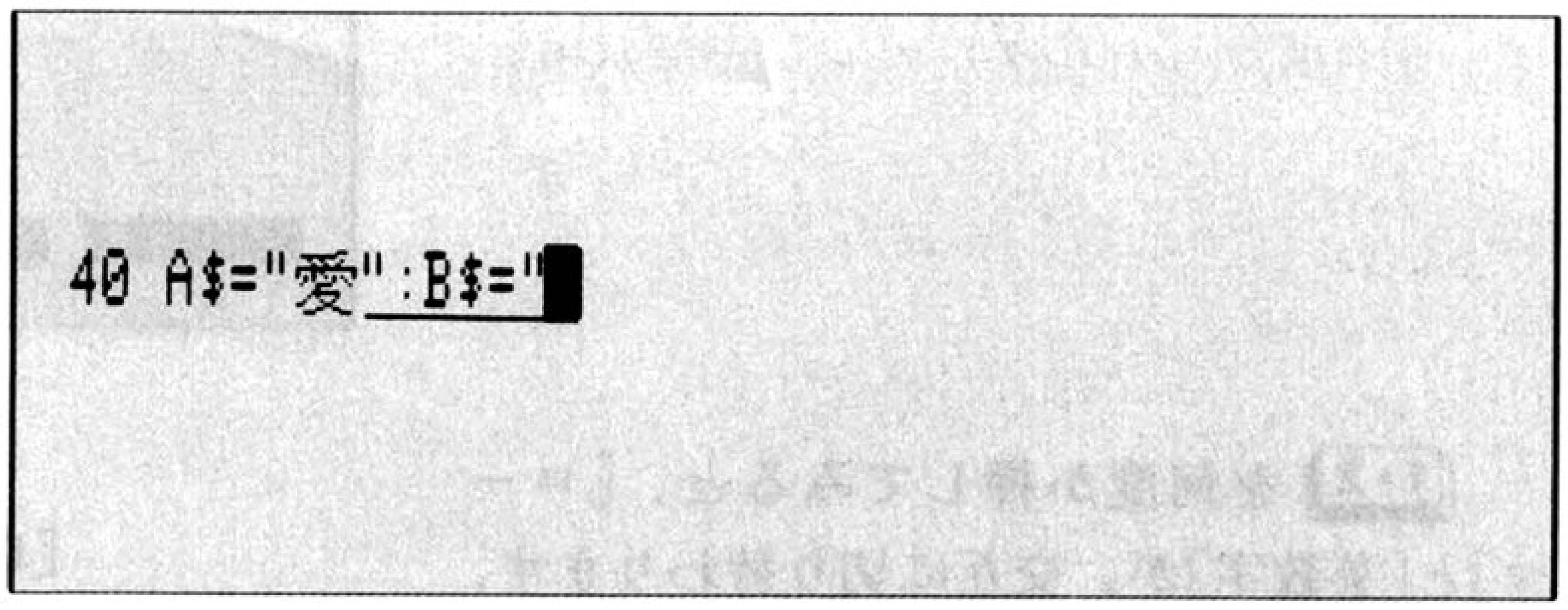
6. 1文字ずつ漢字を入力 f·6 (単 漢 字)

同じ読みの漢字がたくさんあるとき，その中のひとつを f·3 (変 換)を何度も押しながら探すのは大変です．このようなときは， f·6 (単 漢 字)を使います．

f·1 を押して，全角文字で"あい"と入力してから f·6 を押します．「あい」という読みがなの漢字が，画面の最下行に表示されます． → を押すごとにリバース表示が右へ移動していきますから，「愛」のところまで移動し， ↵ を押します．



f·1 を押して，半角文字入力モードに切り替え，続けて入力します(下線の部分)．

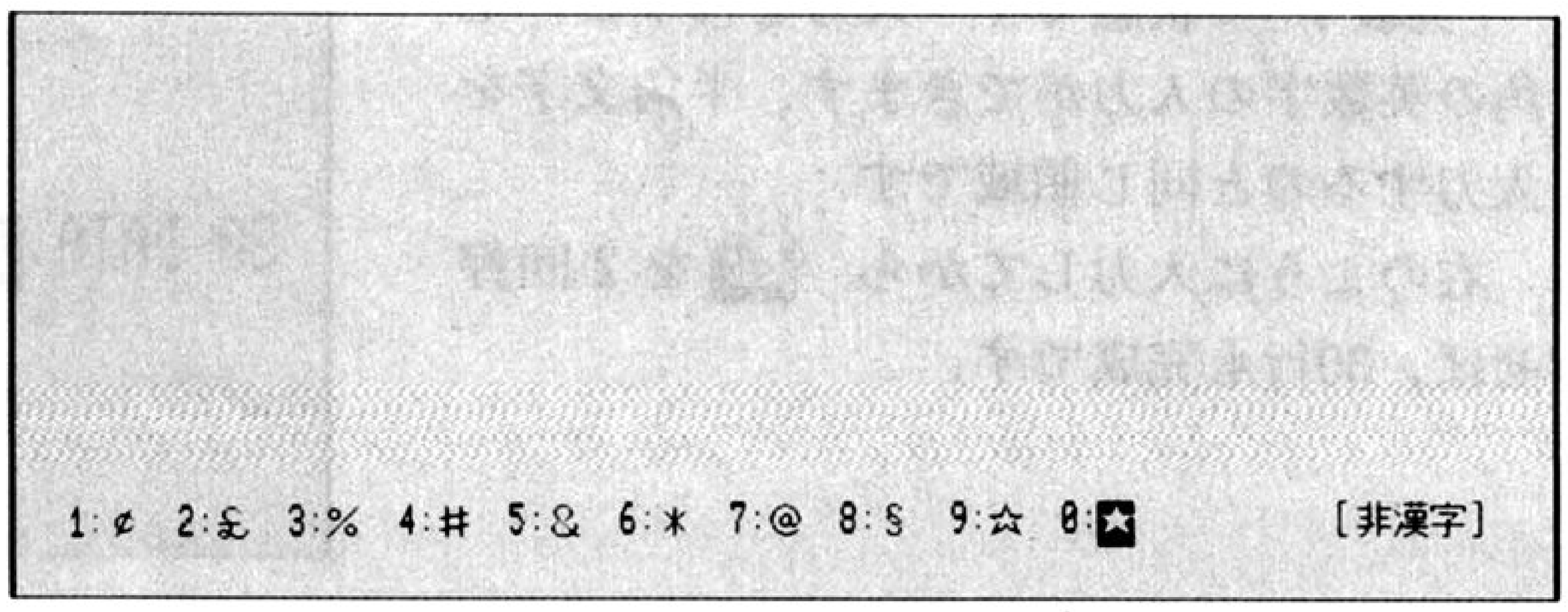


f·1 を押して，全角文字入力モードにしておきます．

7. 特殊文字の入力 f·7 (非 漢 字)

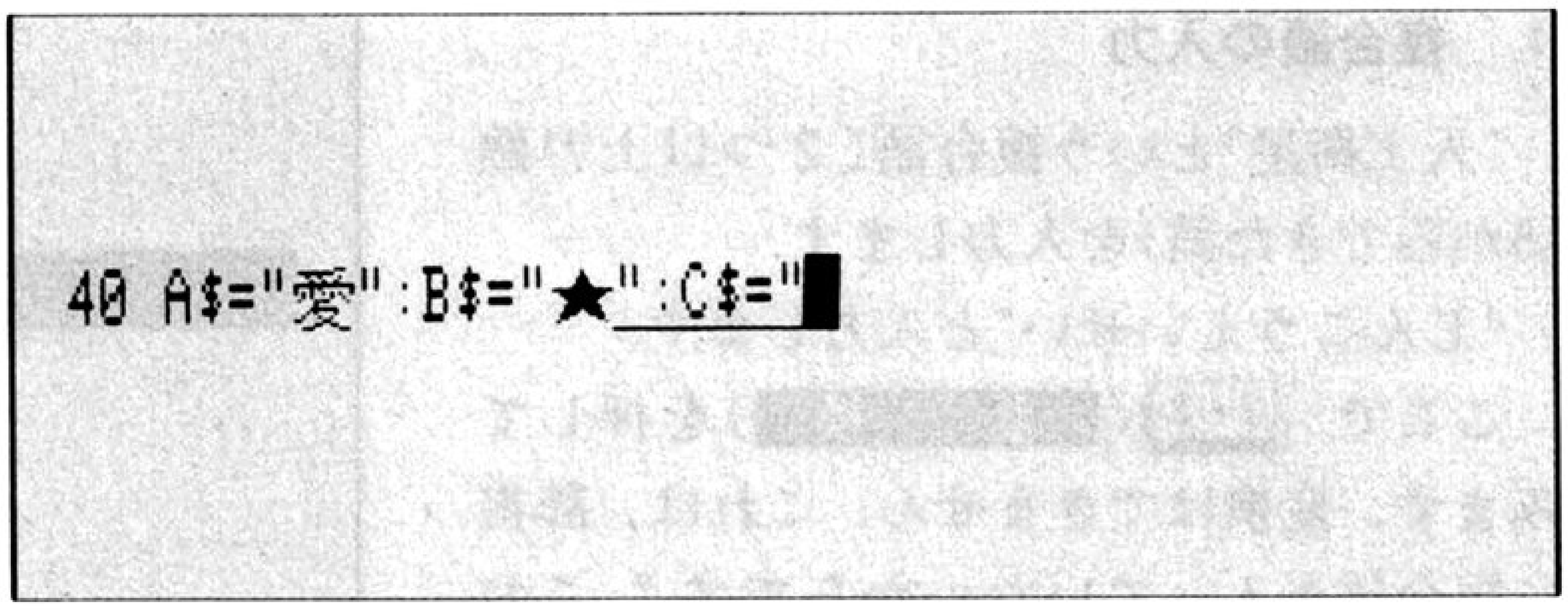
記号，ギリシャ文字，ロシア文字などは f·7 (非 漢 字)を使って入力します．

f·7 を押すと，画面の最下行に候補文字が10個ずつ表示されます． ↑ を押しながら目的の文字を探し， → で目的の"★"を指して ↵ を押します．



f・1 を押して半角文字入力モードに切り替え，下線部分を続けて入力します．

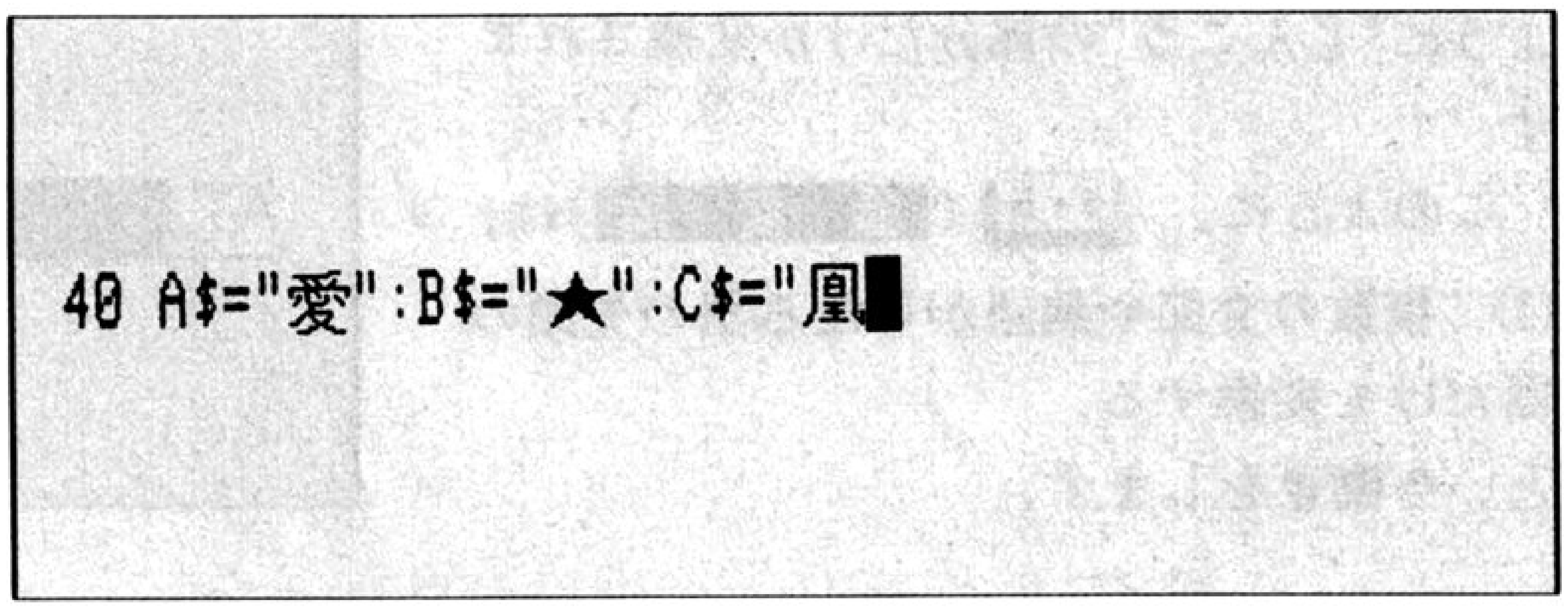
f・1 を押して，全角文字入力モードにしておきます．



8. コード入力
f・9（コ ー ド）

全角文字はJISコード，またはシフトJISコードで入力することもできます．また，JIS第２水準の漢字は，コード入力によって入力することができます．入力したい文字を日本語コード表で探し，そのJISコードまたはシフトJISコードを調べます．ここでは"凰"を入力しますから，JISコードは16進数で5160H，シフトJISコードは16進数で9980Hです． f・9（コ ー ド）を押してから，"凰"を入力したい位置に5160または9980を入力します．４桁目を入力し終わると，文字が現れます．

f・1 を押して半角文字入力モードにして，"（ダブルクォーテーションマーク）を入力し， ↵ を押せばプログラムのできあがりです．



4. 複数の文節／熟語の変換

N88-日本語BASICの日本語入力機能では，複数の文節や熟語を含む文字列を一度に変換することはできません．たとえば，"あいじょう"を"愛情"に，"ひょうげん"を"表現"に変換することはできますが，"あいじょうひょうげん"を一度に"愛情表現"に変換することはできません．

このようなときは， f・5（重 変 換）を使って，複数の文節や熟語の先頭からひとつずつ変換を行います．

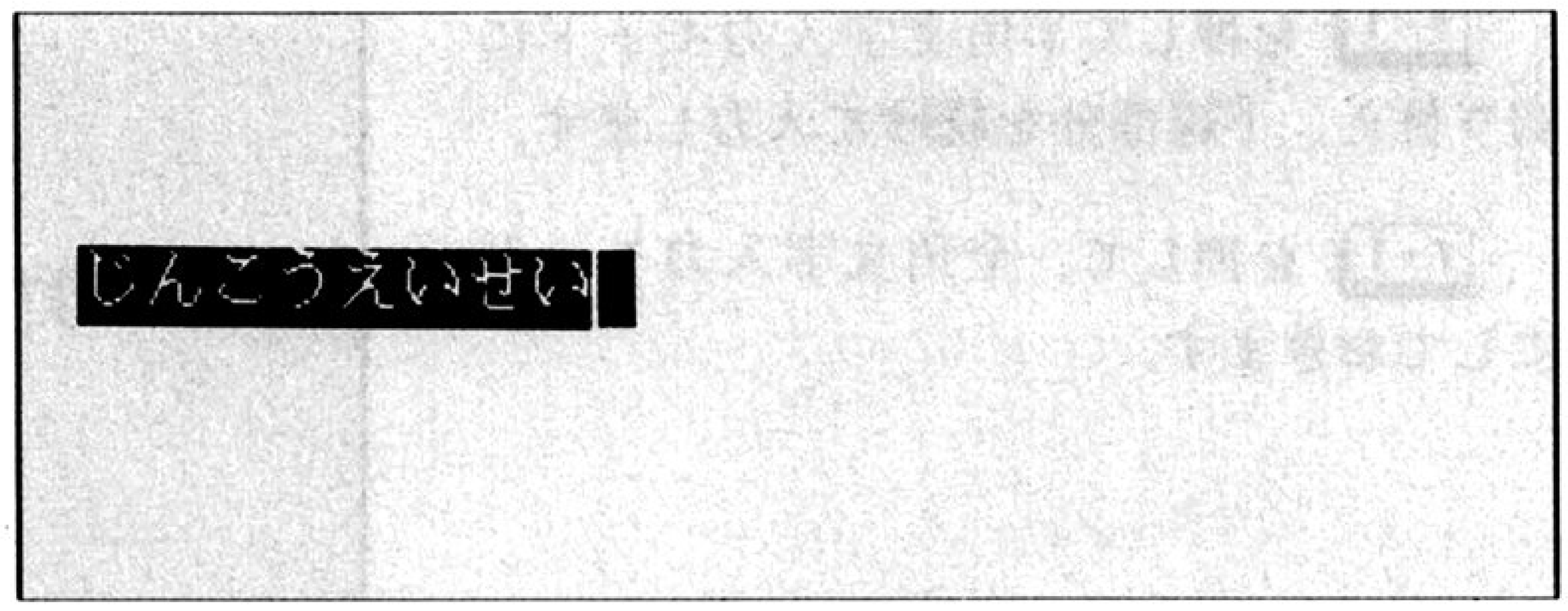
ここでは，３つの例をあげて入力方法を説明します．

1. 複合語の入力

"人工衛星"という複合語(2つ以上の熟語からできた語)を入力します.

"じんこうえいせい"と入力します.

ここで f・3(変　換)を押してみます. 変換はできません. これは, 辞書に複合語が入っていないからです.* このようなときは, f・5(重 変 換)を使います.

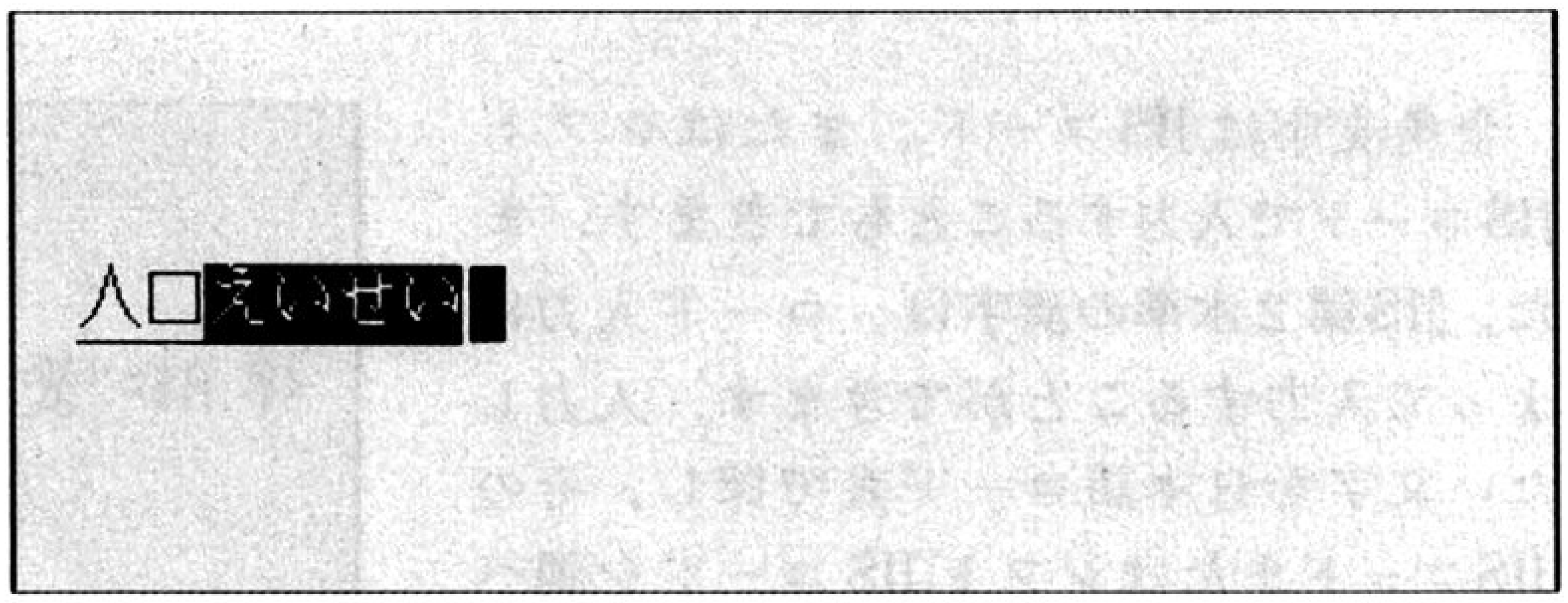


f・5(重 変 換)を押すと, 右のように"じんこう"の部分だけが変換されます.

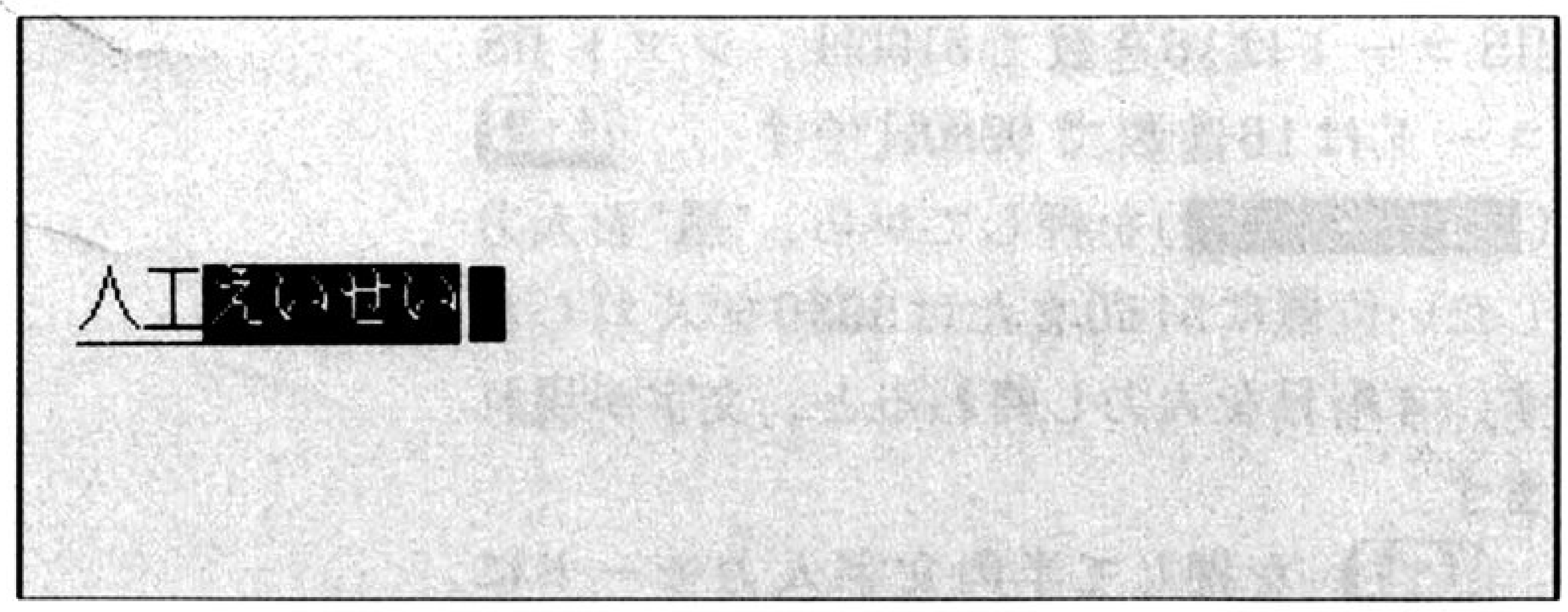
このように, f・5(重 変 換)は,

(1) 複数の文節や熟語からなる語の先頭の語だけを変換する.

という働きをします.



アンダーラインの付いている部分をさらに変換するには, f・3(変　換)を使います. f・3(変　換)を何度か押して希望の語を見つけてください.



アンダーラインの付いている部分が, 現在変換の対象になっています.

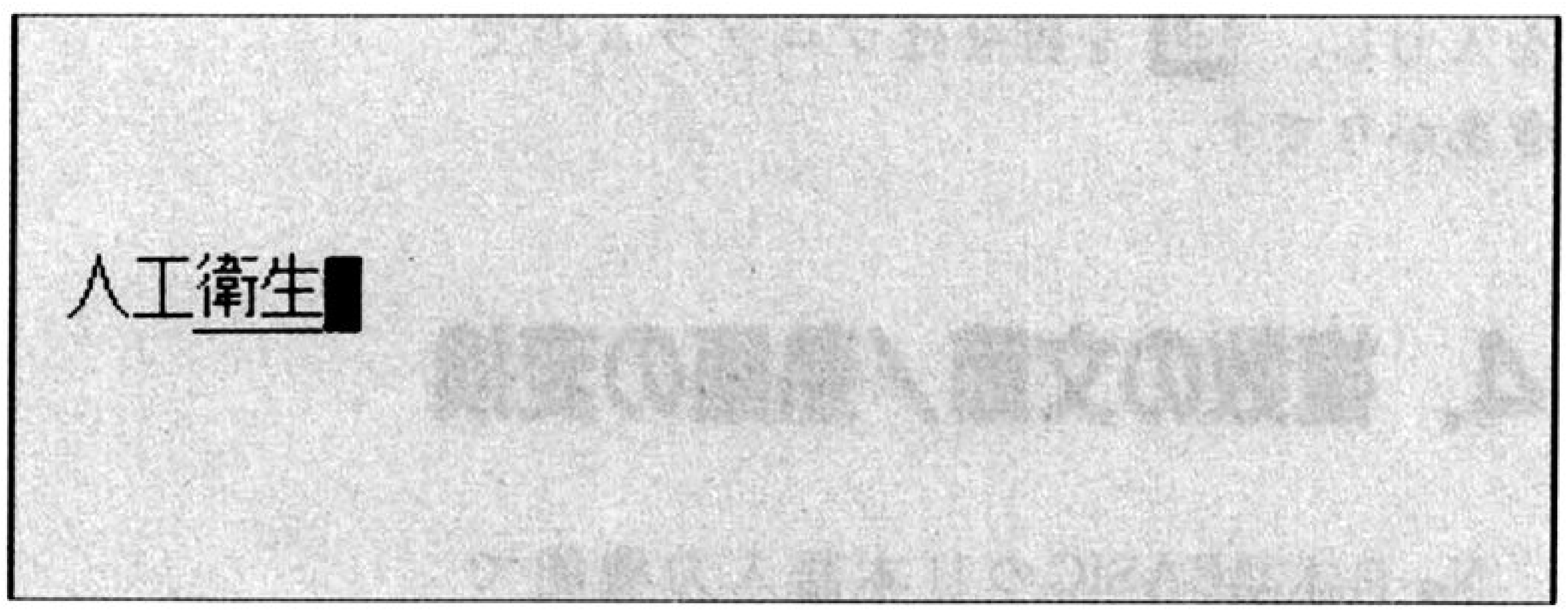
ここで f・5(重 変 換)を押すと, 右のようになります.

このように f・5(重 変 換)は,

(2) アンダーラインの部分を決定する.

(3) リバース表示の部分を変換する.

という働きをします.



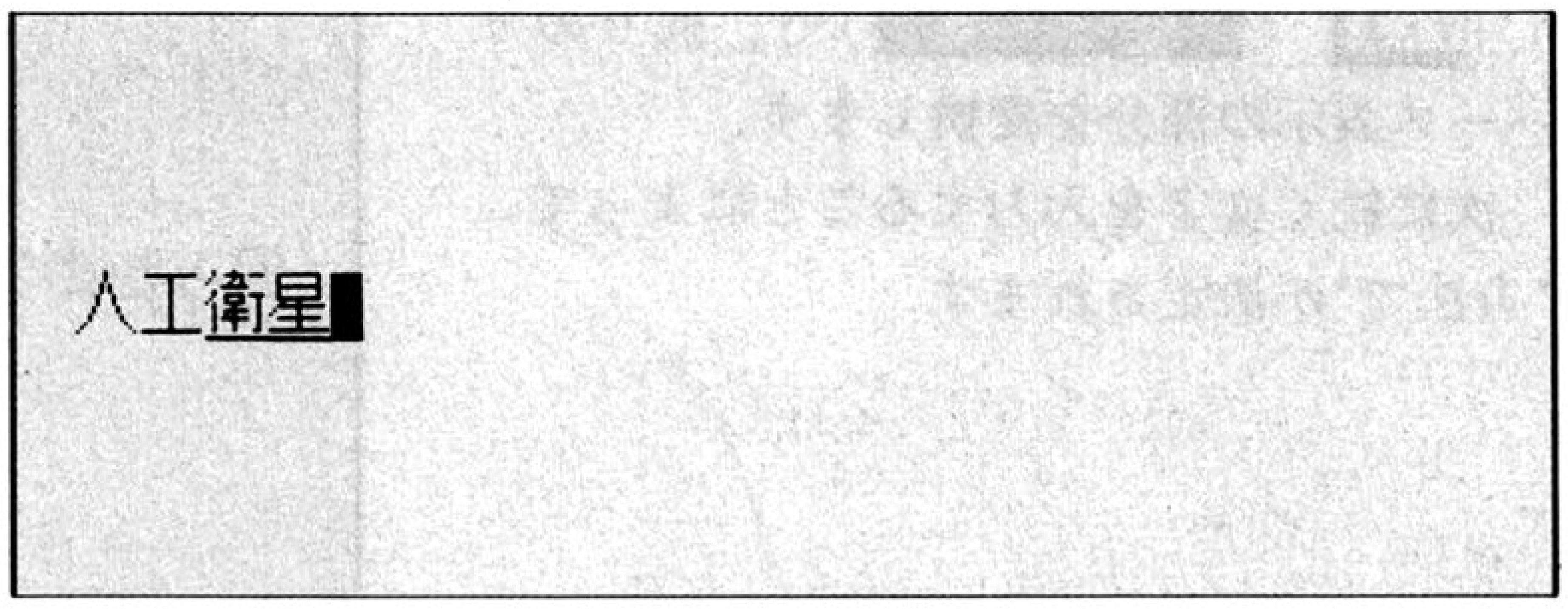
* N88-日本語BASICでは, 熟語に, 接頭語[1], 付属語[2]を付けた, 文節単位で一度に変換を行いますから, たとえば"いがくはくし"を, f・3(変　換)によって変換すると, "は"を付属語と解釈して"医学は"という文節に変換してしまいます.

1) "あなたの", "あらゆる", "この", "あの", "お", "御", "他の", "約"などの体言の前に付く語を接頭語といいます.

2) "て", "に", "を", "は", "と", "も", "の"などの体言の後に付く語を付属語といいます.

アンダーラインの部分をさらに変換するには，f・3（変　換）を押します．

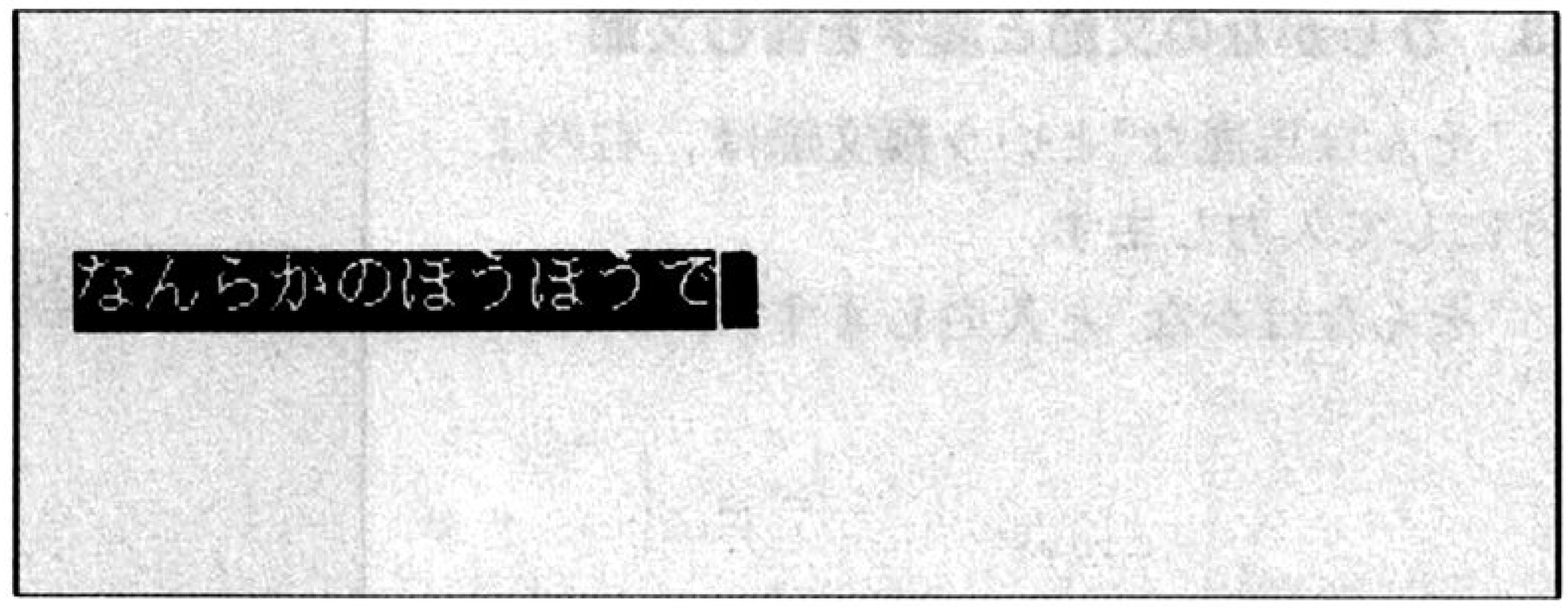
次に続く文字を入力することによって，"衛星"が決定されます．



2. 複文節の入力

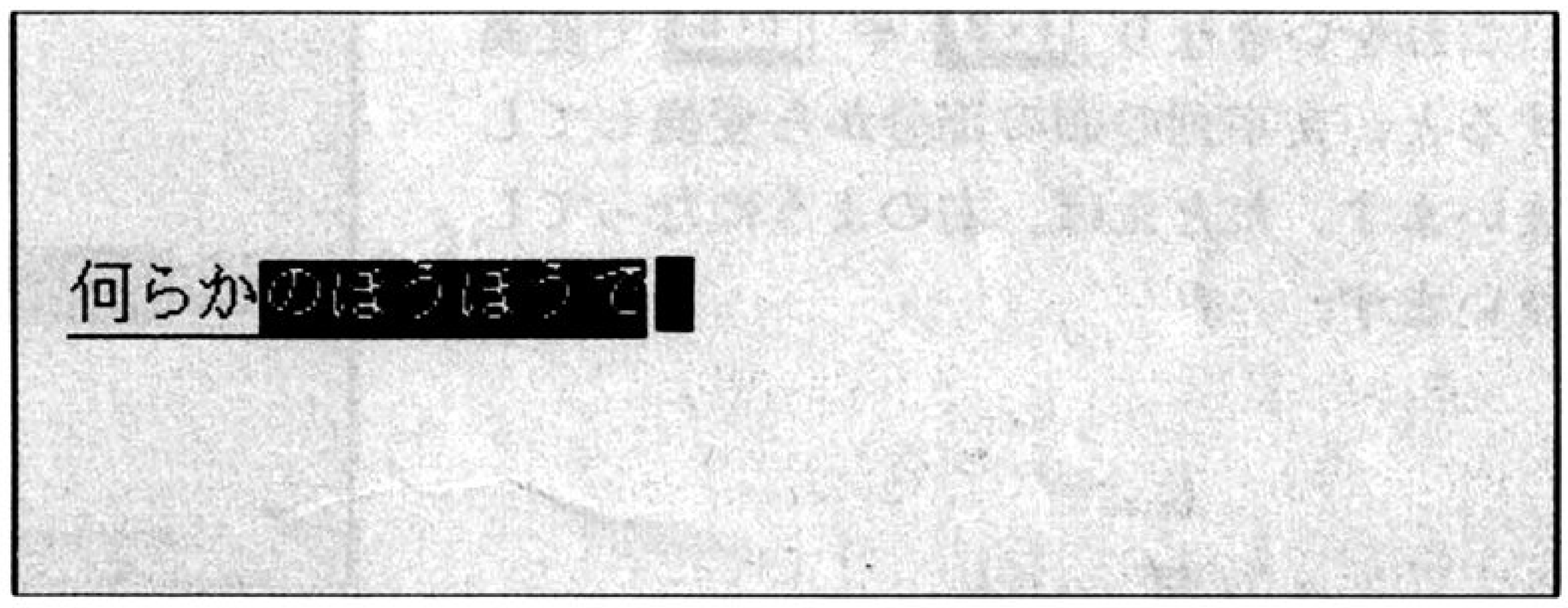
"何らかの方法で"という，2つの文節からなることばを入力します．

"なんらかのほうほうで"と入力します．



f・5（重　変　換）を押す*と，右のように"何らか"の部分が変換の対象になります．

ここで f・5 を押してしまうと，"のほうほうで"の部分が変換の対象になってしまいます．"の"の部分をひらがなのままにしておくには，SHIFT ＋ → を使います．



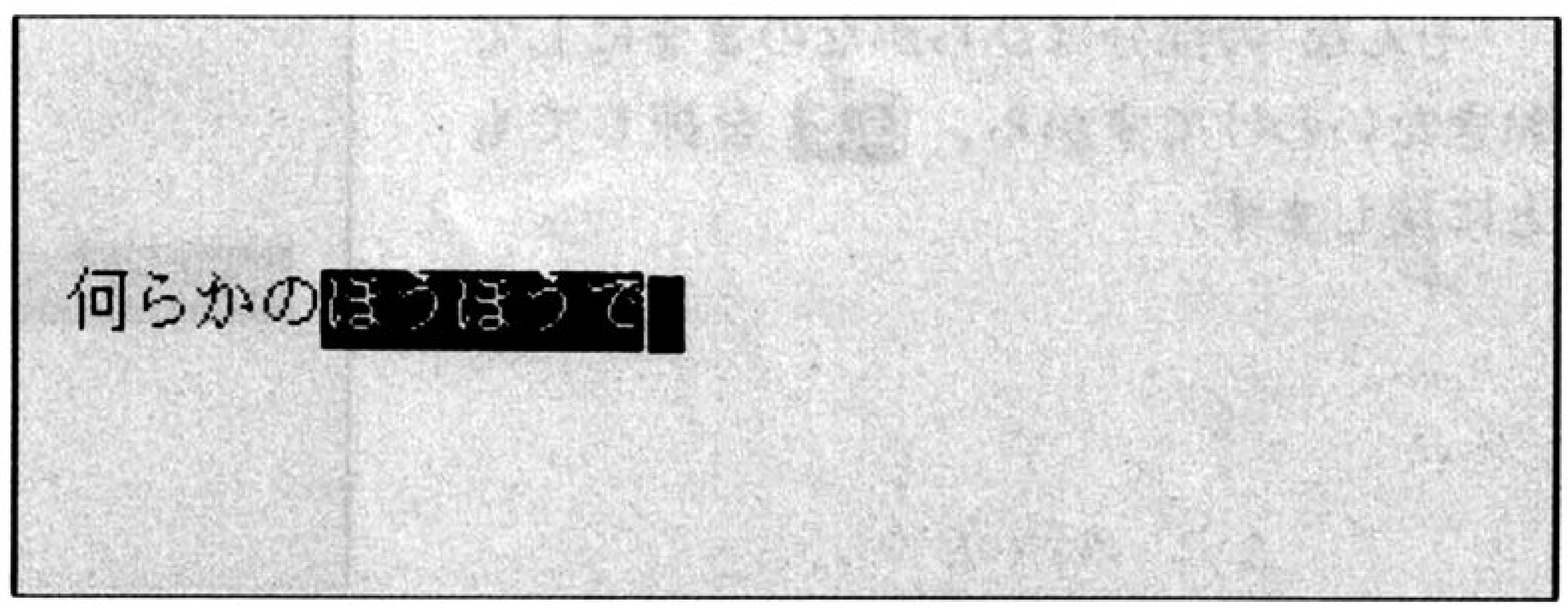
SHIFT ＋ → を押すと，右のようになります．

このように SHIFT ＋ → は，

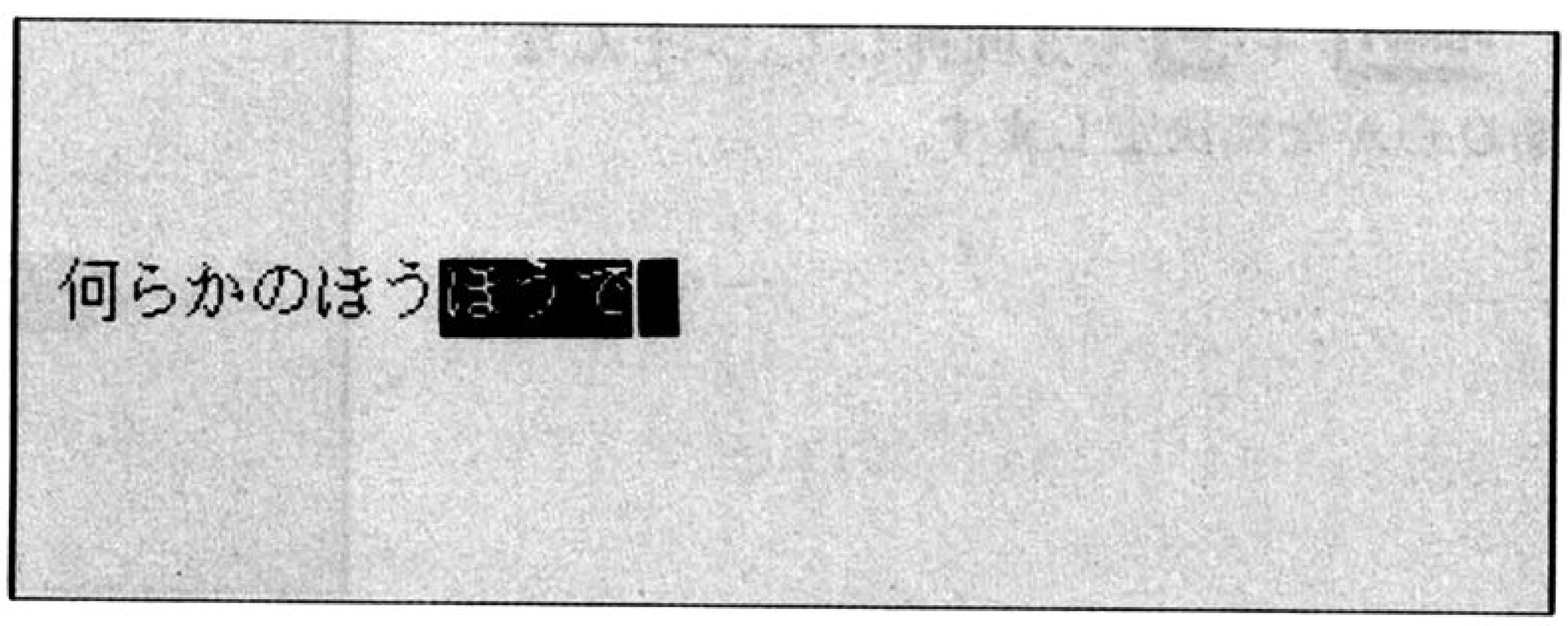
(1)　アンダーラインの部分を決定する．

(2)　リバース表示になっている部分を，1文字ずつ決定する．

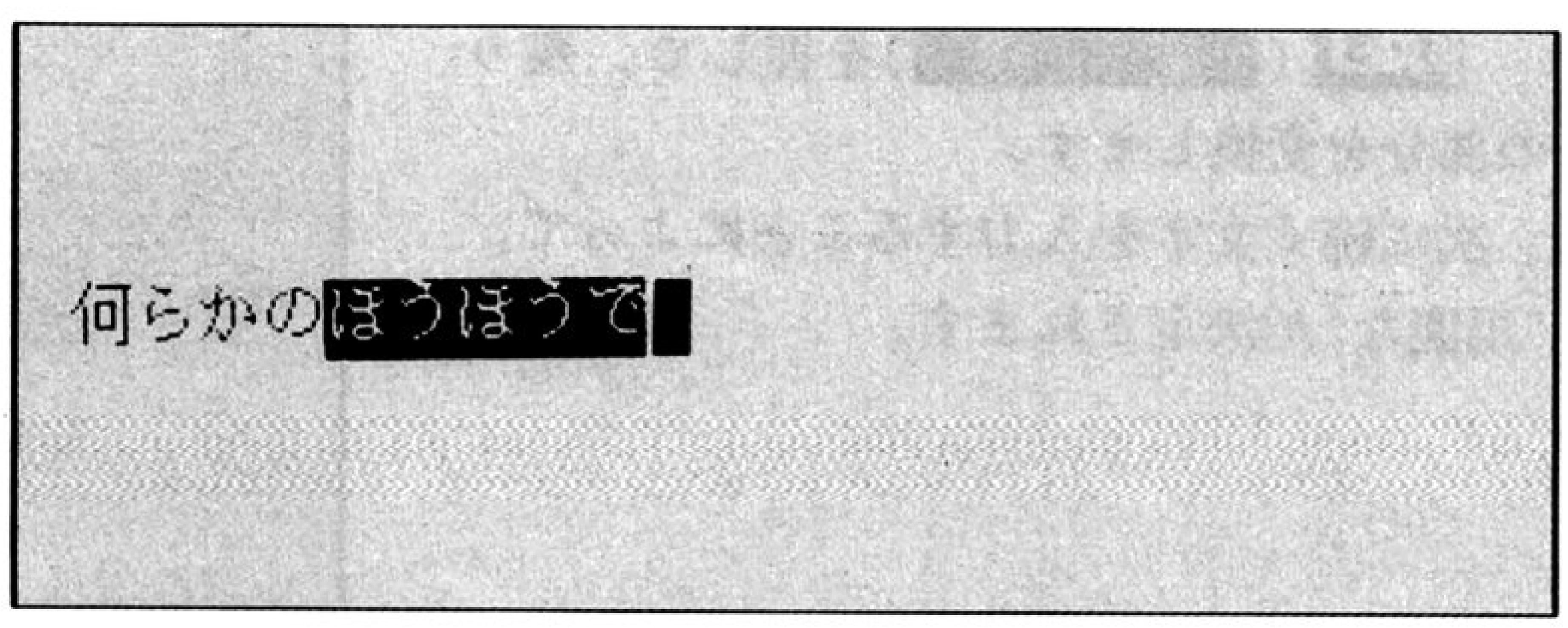
という2つの働きをします．



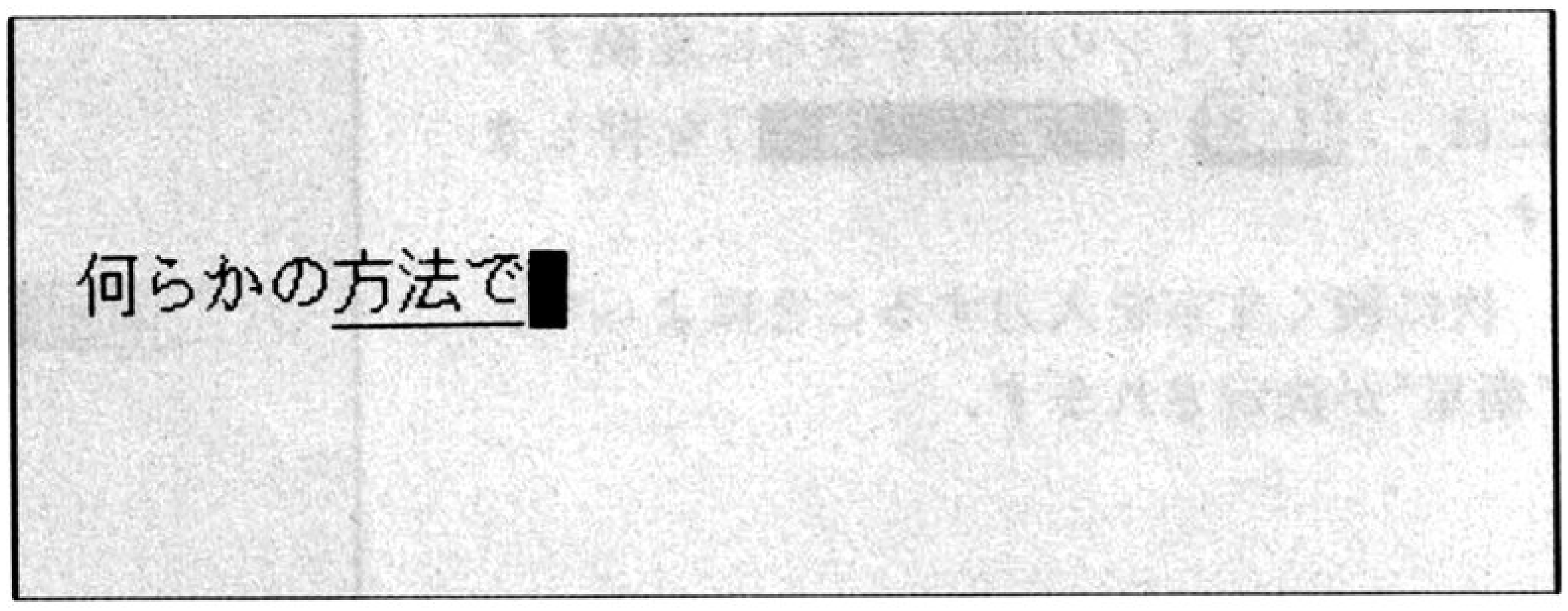
誤って SHIFT ＋ → を押し過ぎてしまい，決定した部分が長くなってしまった場合は，SHIFT ＋ ← で1文字ずつもとに戻すことができます．たとえば，右のようになってしまった状態で SHIFT ＋ ← を2回押せば，もとに戻ります．



＊ この場合は，f・3（変　換）を使うこともできますが，複数の文節や熟語からなる語の変換は，原則として f・5（重　変　換）を使うようにしてください．

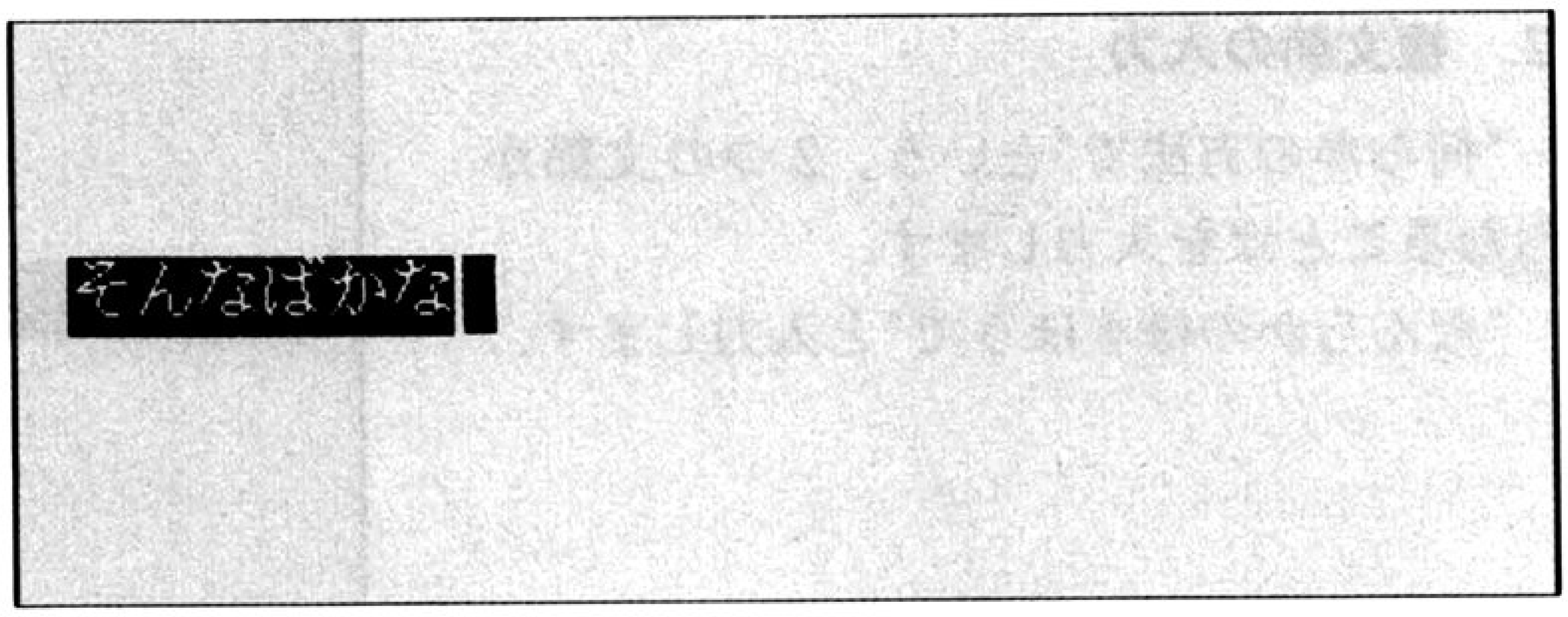


f·3（変　換）で，残りのリバース表示の部分を変換します．
次に続く文字を入力することによって，"方法で"が決定されます．

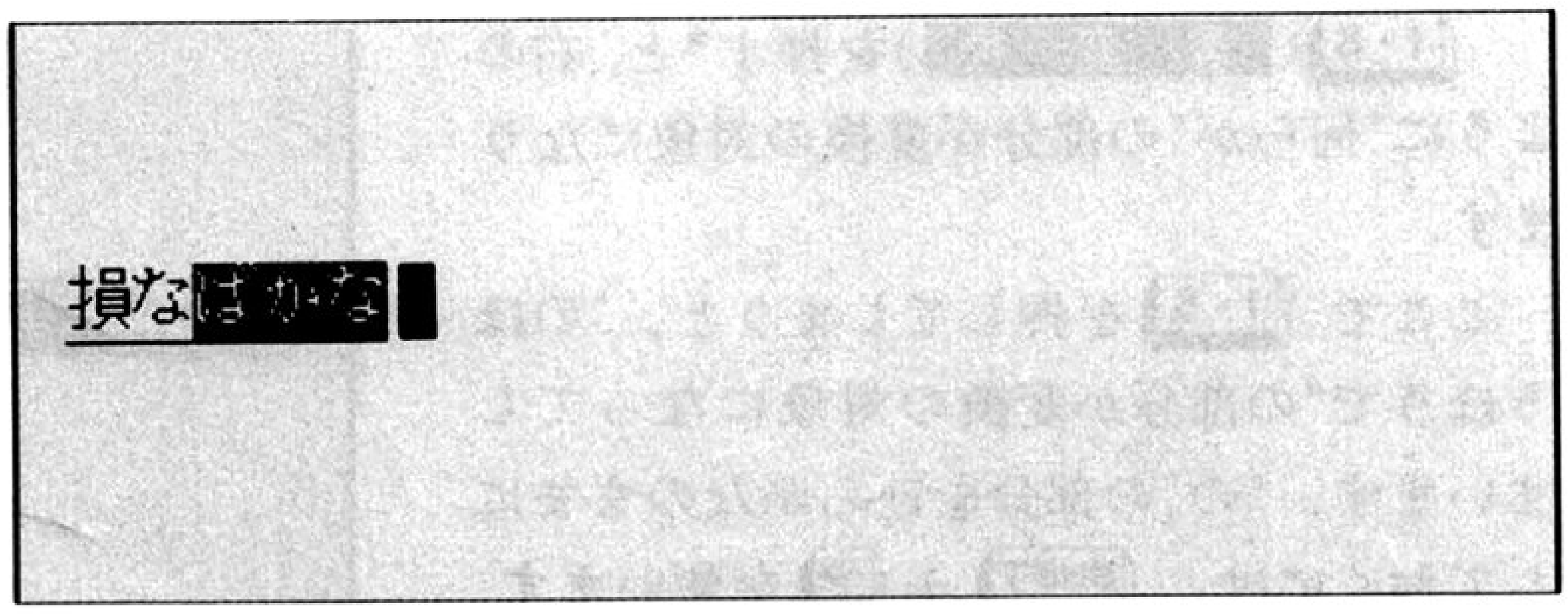


3. ひらがなの文節と漢字を含む文節

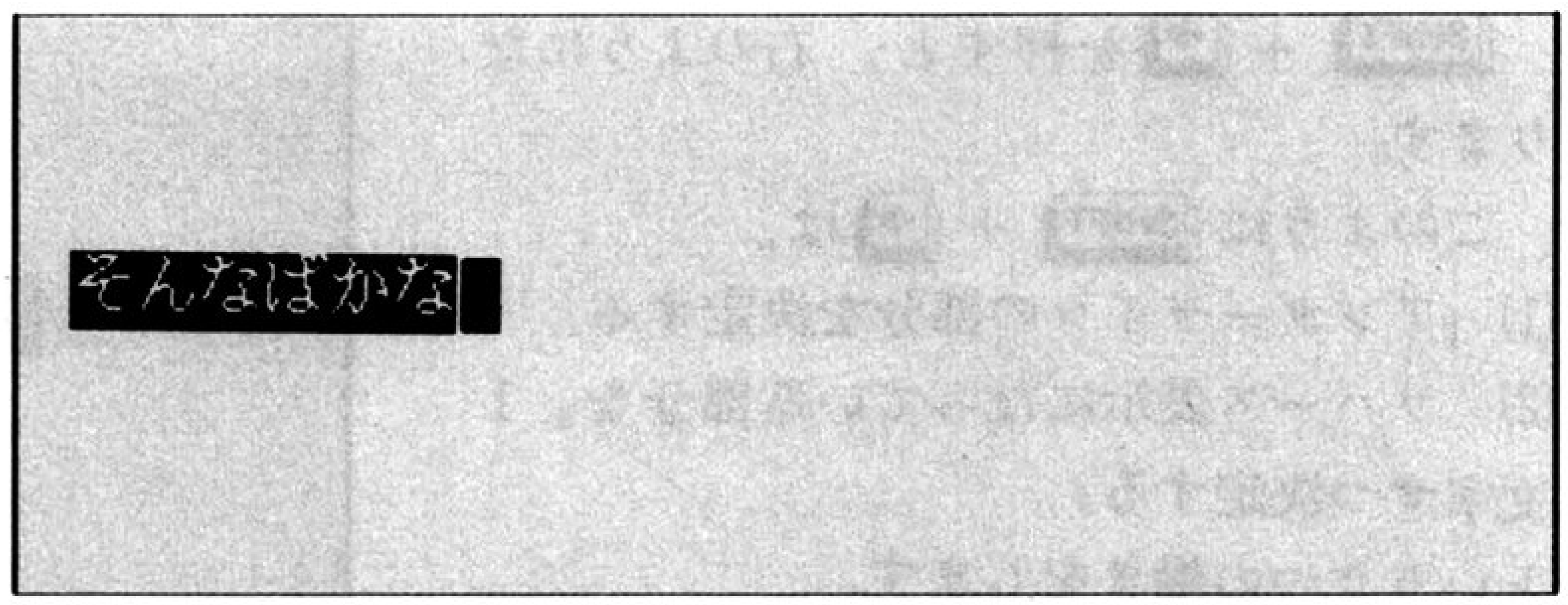
"そんな馬鹿な"という複文節は，右のようにして入力します．
"そんなばかな"と入力します．



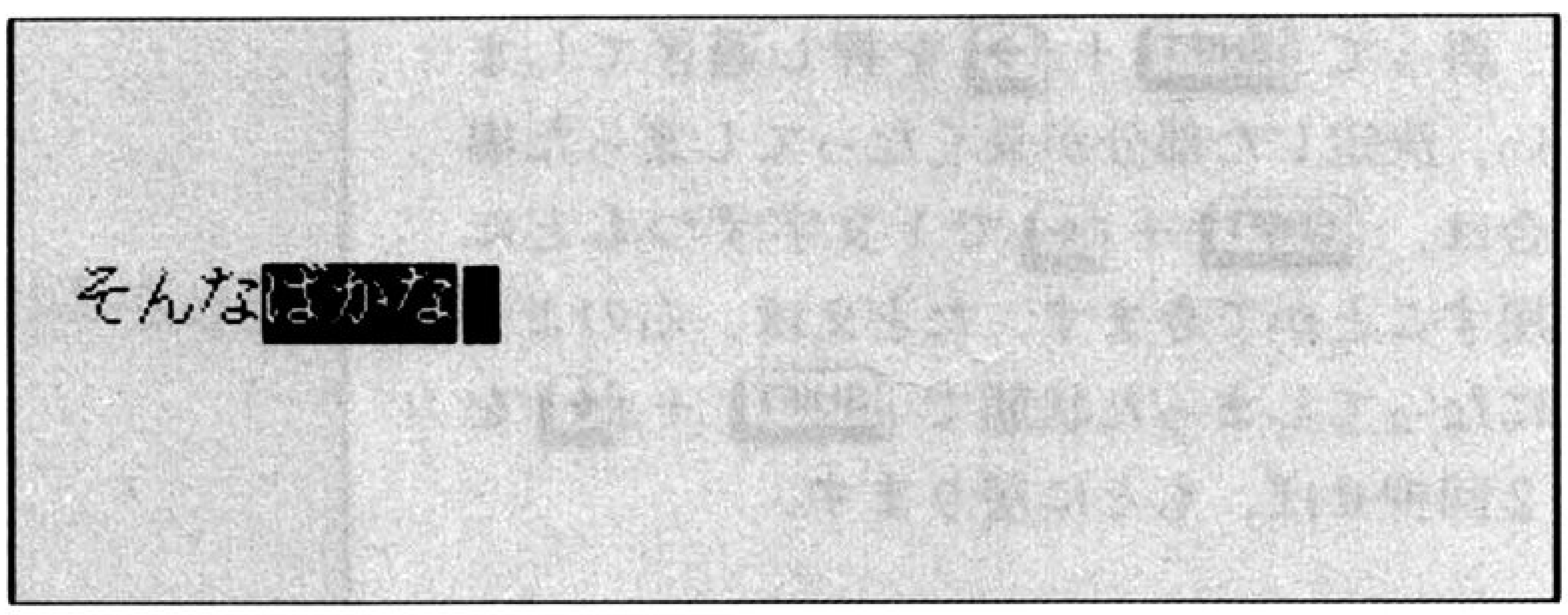
これをいきなり f·3 や f·5 で変換すると，文字列の頭の部分から変換してしまいます．たとえば，右のようになってしまいます．



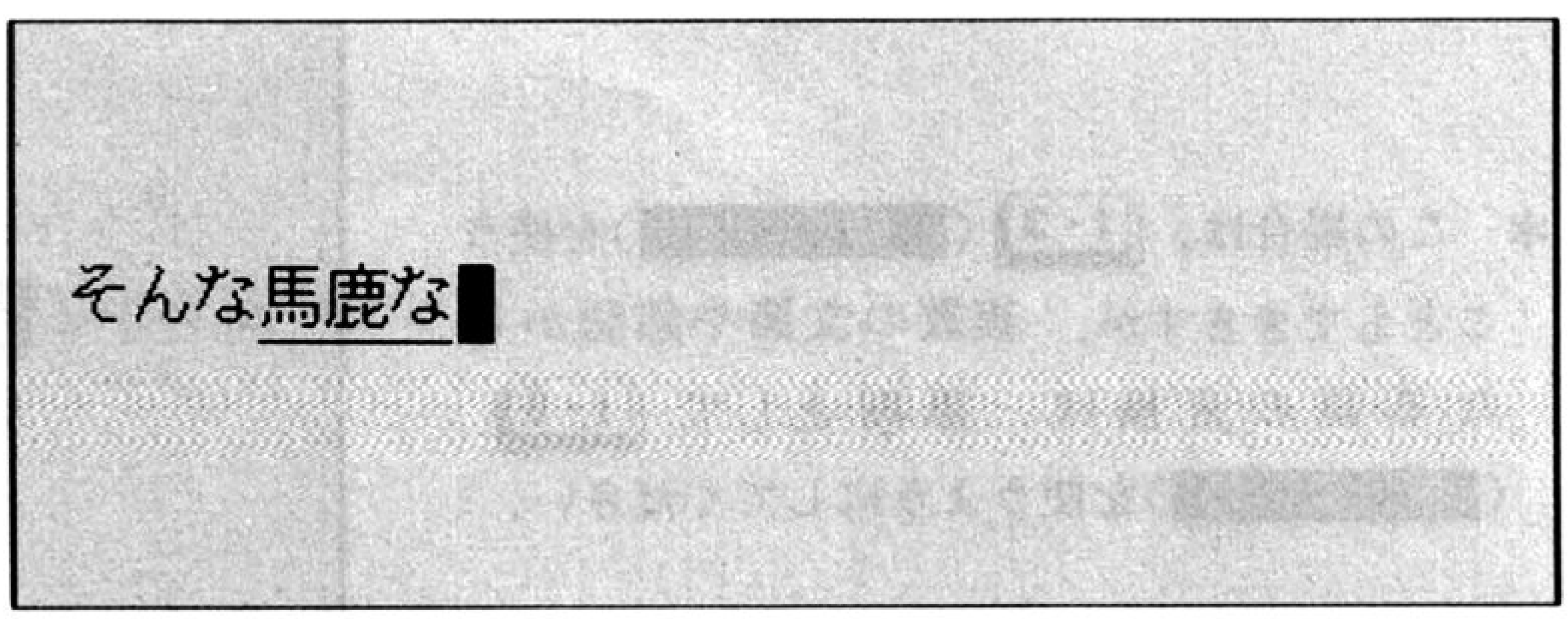
"そんな"の部分はひらがなのままにしておきたいわけですから，ESC を押してもとに戻します．



SHIFT ＋ → を3回押して，"そんな"をひらがなに決定します．



f·3（変　換）を押して，残りの部分を変換します．
次に続く文字を入力することによって，"馬鹿な"が決定されます．

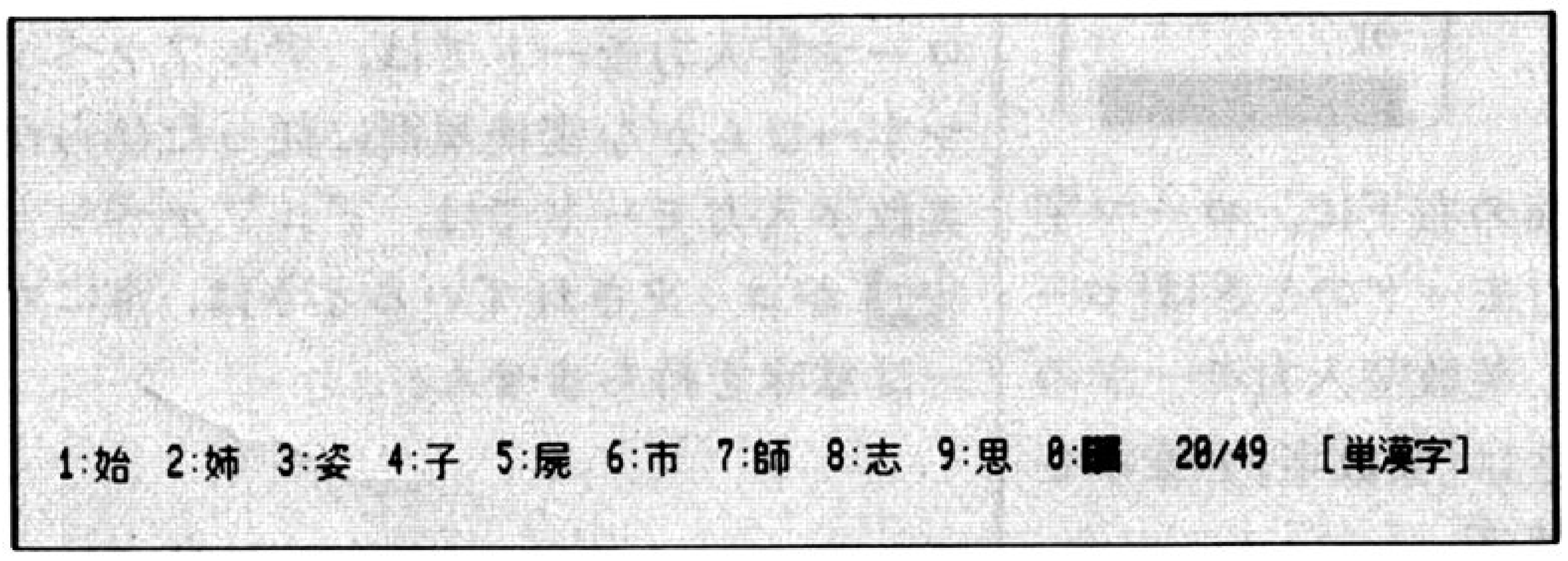
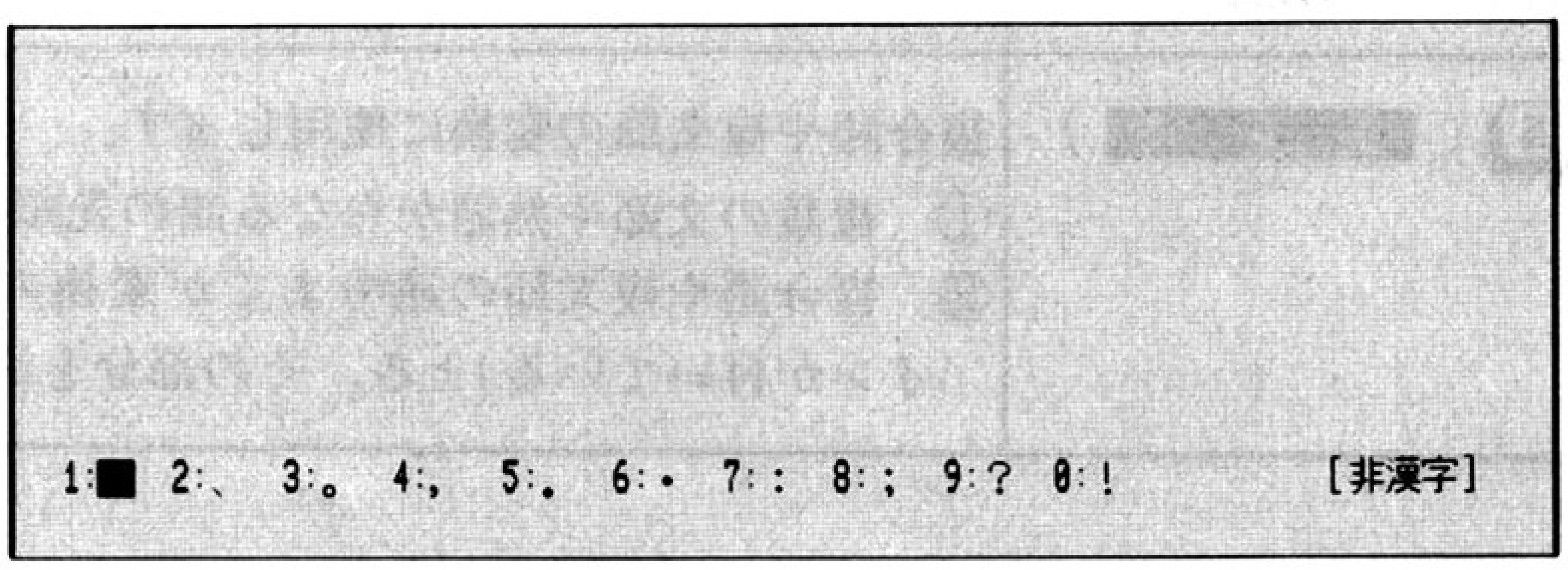


5. 各キーの使い方

日本語文字の基本的な入力方法は，前節までのでひととおり説明してきました．

ここでは，日本語の入力に慣れた方が参照できるように，各キーごとにその使い方をまとめます．

キー	機能
f·1 （全角／半角 or 全 半）	全角文字の入力モードと半角文字の入力モードとの切り替えを行います． 全角文字のモードでは，f·2 ～ f·10 がこの表に示されている内容になります．
f·2 （ローマ字 or ロ ー マ） 画面の右下に，ローマ字入力モードのときは[ローマ]，英数字入力モードのときは[英数字]と表示されます．	ローマ字入力モードと英数字入力モードの切り替えを行います． ローマ字入力モードでは，アルファベット(A～Z)を入力すると，ローマ字→ひらがな変換規則に従ったひらがなに変換されます． 英数字入力モードでは，アルファベットはそのまま入力されます． カナ がロックされているときは，常にカナ入力になりますので，このキーは意味を持ちません．
f·3 （変 換） または スペース	リバース表示されている，またはアンダーラインのついた範囲の文字列を，文節単位に漢字に変換します． このキーを繰り返し押すことにより，次々に漢字候補が現れます． 候補は次のいずれかの動作によって決定されます． (1) 次の文字を入力する． (2) ⏎ または f·1 を押す． 単漢字変換(f·6)の途中からでも，このキーを押すことによって，この変換に切り替えることができます．
f·4 （無 変 換） または SHIFT ＋ スペース	入力された文字列を，変換せずにそのまま決定します．また，変換中の文字列を入力したときの状態に戻します．
f·5 （重 変 換）	複合語や複文節の変換に使用します． ① 複数の文節や熟語からなる語の先頭の語だけを変換します． ② 複合語や複文節の途中までが変換の対象になっている(アンダーラインが付いている)とき，その部分を決定し，それに続く語を変換します．

キー	機能
[f・6] (単漢字) 単漢字の選択中は，画面右下に[単漢字]と表示されます．	変換の対象の読みを持つ，1文字の漢字に変換します． また，次のような読みを漢字以外の文字に変換します． (1) 「ぎりしあ」→[f・6]→ギリシャ文字 (2) 「ろしあ」 →[f・6]→ロシア文字 (3) 「かな」 →[f・6]→特殊かな(ゎゐゑヮヰヱヵヶ) (4) 「がいじ」 →[f・6]→外字 [f・6]を押すと，画面の最下行に，10文字ずつ候補文字が表示されます(40文字/行のときは5文字ずつ)．11文字以上の候補がある場合は，[↑]，[↓]によって次の候補群，前の候補群を表示させます．[単漢字]の左の数字によって，全部で何文字の候補があり，その中の何番目の候補が表示されているかがわかります．たとえば次のように表示されているときは，49文字のうちの20番目の候補を指しているということになります．  候補は次の方法のいずれかで決定します． (1) 0～9の数字を入力する． (2) [→]，[←]で文字を選んで[⏎]を押す．
[f・7] (非漢字) 非漢字の選択中は，画面右下に[非漢字]と表示されます．	この機能によって，次の文字を入力します． (1) 記号(JISコード 2121H～222EH) (2) ギリシャ文字 (3) ロシア文字 (4) 特殊かな(ゎゐゑヮヰヱヵヶ) (5) 外字 [f・7]を押すと，画面の最下行に候補文字が表示されます． ただし，未決定の文字があるときは非漢字の入力をすることはできません．  候補の選択方法は，単漢字と同じです．
[f・8] (前候補)	[f・3] (変換)によって，候補文字列を選択している途中で，[f・8]を押すことによって，1つ前の候補に戻すことができます． 同じ読みを持つ候補がたくさんあるときに誤って，[f・3]を押しすぎたときに，[f・8]で前の候補に戻します．

キー	機能
f・9 （コード） コードの入力中は画面右下に[コード]と表示されます.	JISコードまたは，シフトJISコードを16進数で指定することによって，全角文字を入力します. f・9 を押した後，4桁目の16進数を入力します(3桁目までで，入力を誤ったときは INS DEL によって修正することができます)．4桁目の入力と同時に，そのコードで表される全角文字が表示されます. ただし，未決定の文字があるときはコードの入力はできません.
f・10 （カタカナ or カナ）	リバース表示されている，変換対象文字列をカタカナに変換します. カタカナに変換したあとで，f・4（無変換）によって，ひらがなに戻したり，f・3（変換）によって，漢字へ変換を続けることができます.
⏎	未決定の文字がある場合は，⏎ によって決定します. 未決定の文字がない場合は，改行を行います.
ESC	ESC は以下の２つの働きをします. (1) f・3（変換），f・5（重変換），f・10（カタカナ）で表示される候補(リバース表示，または下線つきで表示されているもの)を，入力直後の状態(ひらがな)に戻します. (2) 単漢字入力（f・6），非漢字入力（f・7），およびコード入力（f・9）の途中で，入力を中断します.
カナ ロックされていると，画面の右下に[か　な]と表示されます.	カナ をロックすると，カナ入力モードになり，各キーに記されているカタカナに相当するひらがなが入力できます. ロックを外すと，ローマ字でのひらがな入力または，英数字の入力モードになります.
SHIFT ＋ →	複分節の語の変換の際，現在変換の対象になっている語を決定し，それに続く文字を１文字ずつ決定します．または，SHIFT ＋ ← によって未決定にした文字列の左端から１文字ずつ決定します.
SHIFT ＋ ←	SHIFT ＋ → などによってひらがな，カタカナ，英数字に決定した部分を１文字ずつ，もとの未決定の状態に戻します.
GRPH ＋ スペース	全角のスペースが入力されます. スペースの左隣の文字列が変換の途中だった場合は，スペースの入力によって決定されます.
← →	変換中の文字列を決定し，カーソルを左右に移動します.

6. 注意事項

(1) CONSOLE文の第3パラメータ〈ファンクションキー表示スイッチ〉を0に指定すると，ファンクションキーに登録されている文字列は，画面に表示されません．

この状態では，全角文字を入力することはできません．全角文字を入力する必要のあるときは，必ずCONSOLE,,1を実行してください．

(2) 変換途中の文字列（リバース表示，または下線付きになっている文字列）があるときは，↑，↓によってカーソルを上下に動かすことはできません．

(3) 単漢字と非漢字の文字候補の語順には，学習機能がありません．

第3章
日本語処理 (N88-日本語BASIC)

PC-8801mkIIMRは，JIS第1,2水準の日本語文字を格納したROMを標準で備えています．
また，プログラミング言語としてN88-BASICに，日本語処理機能を付加したN88-日本語BASICを用意し，強力な日本語処理機能を持っています．
ここでは，PC-8801mkIIMRの持つ日本語処理機能をより有効にお使いいただくために説明をしていきます．
注意：N88-日本語BASICは，フロッピーディスクドライブがない場合は使用できません．

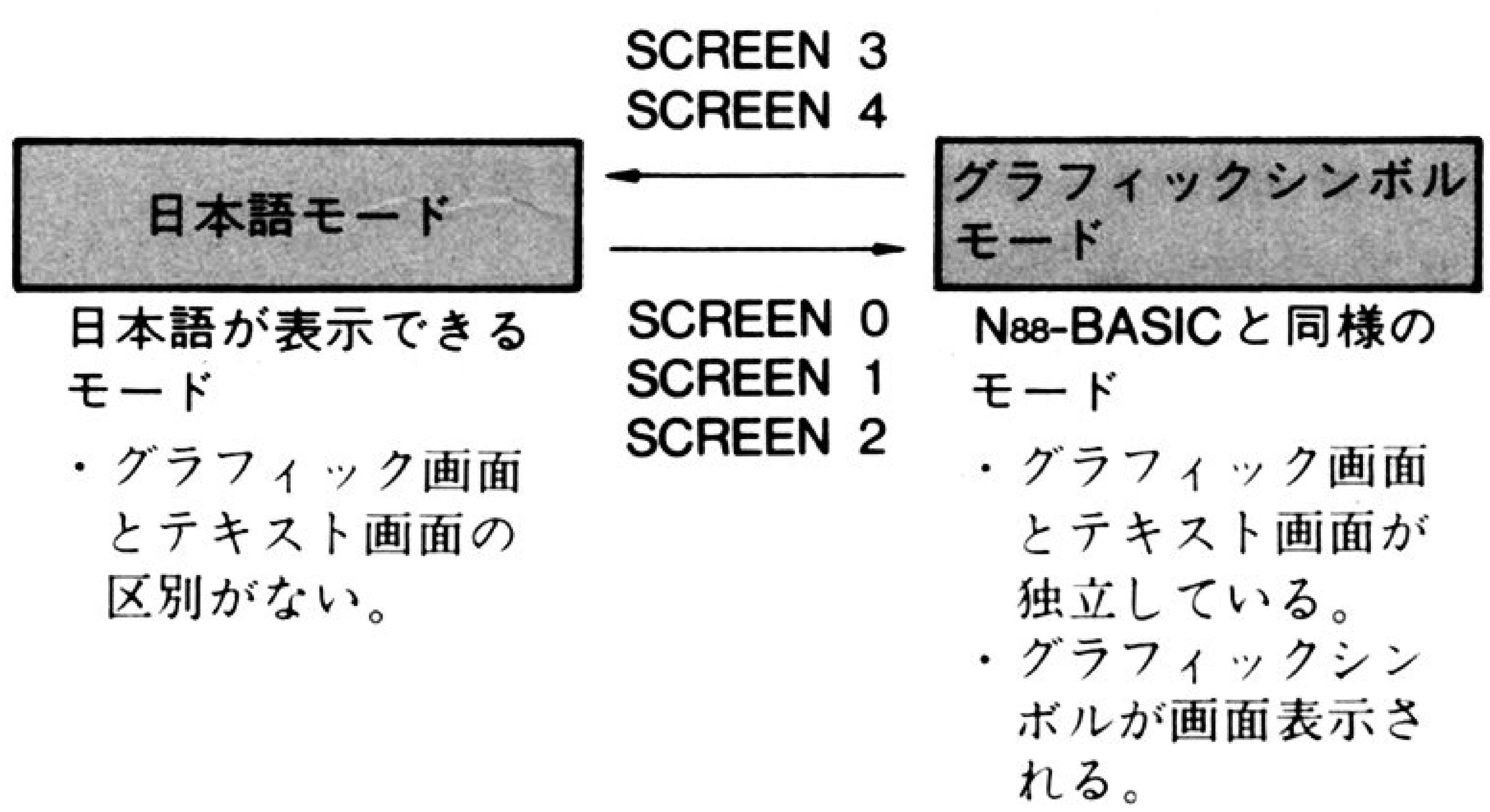
1．モード設定

日本語を扱うときは，日本語を表示できるモード("日本語モード")にしておく必要があります．N88-日本語BASICをスタートすると，初期値として日本語モードになっています．

これに対して，N88-BASICと同じように日本語は表示せず，テキスト画面とグラフィック画面とを区別して持つモードがあります．こちらをグラフィックシンボルが表示できることから"グラフィックシンボルモード"と呼びます．

2つのモードはSCREEN文で切り替えます．右に2つのモードの切り替え方法とそれぞれの特長を図示します．

画面モードに関して，詳しくはBASICリファレンスマニュアルをご覧ください．



2. 日本語文字(全角文字)

2.1 半角文字と全角文字

BASICでは，PRINT文，INPUT文，REM文や文字定数，文字変数(変数名に$が付くもの)，STRING$，MID$などの関数によって，さまざまな文字列を扱います．

N88-BASICなどの日本語処理機能を持たないBASICでは，文字列として資料5のキャラクタコードにあげられている256文字(半角文字)だけを扱います．

これに対してN$_{88}$-日本語BASICでは，この半角文字に加えて日本語コード（BASICリファレンスマニュアル資料８）にあげられている6988種類の全角文字（6535種類の漢字と，453種類の漢字以外の文字）を扱うことができます．

2.2 シフトJISコード

N$_{88}$-日本語BASICは，全角文字を"シフトJISコード"と呼ばれるコードで扱います．

日本語文字は，一般的にはJISの規格で定められている"JISコード"と呼ばれるコードで扱われていますが，N$_{88}$-日本語BASICでは，内部で処理のしやすいシフトJISコードを採用しています．

したがって，外部の機器との間で日本語データをやり取りする場合や，N$_{88}$-日本語BASICで作ったファイルを他のコンピュータが読み取る場合などは，シフトJISコードをJISコードに変換する必要があります．

シフトJISコードとJISコードの変換のサブルーチンの例をあげておきます．

JISコード→シフトJISコード変換サブルーチン

（JIS$にJISコードを入れておくと，SHIFTJIS$にシフトJISコードを入れて返すサブルーチンです．）

```
100 'JIS→シフトJIS  コード変換サブルーチン
110 JH=VAL("&h"+LEFT$(JIS$,2))                  ' JH:JISコードの1バイトめ
120 JL=VAL("&H"+RIGHT$(JIS$,2))                 ' JL:JISコードの2バイトめ
130 IF JH<=&H5E THEN SJH=INT((JH-1)/2)+&H71
                ELSE SJH=INT((JH-1)/2)+&HB1     'SJH:シフトJISコードの1バイトめ
140 IF JH MOD 2=0 THEN SJL=JL+&H7E:GOTO 160
                  ELSE SJL=JL+&H1F              'SJL:シフトJISコードの2バイトめ
150 IF SJL>=&H7F THEN SJL=SJL+1
160 SHIFTJIS$=HEX$(SJH)+HEX$(SJL)               'SHIFTJIS$:シフトJISコードを
                                                 16進数の文字列で返す
170 RETURN
```

シフトJISコード→JISコード変換サブルーチン

（SHIFTJIS$にシフトJISコードを入れておくと，JIS$にJISコードを入れて返すサブルーチンです．）

```
100 'シフトJIS→JIS  コード変換サブルーチン
110 SJH=VAL("&H"+LEFT$(SHIFTJIS$,2))
120 SJL=VAL("&H"+RIGHT$(SHIFTJIS$,2))
130 IF SJL<=&H9E THEN GOSUB *LESSER9EH ELSE GOSUB *GREATER9EH
140 JIS$=HEX$(JH)+HEX$(JL)
150 RETURN
160 *LESSER9EH
170 IF SJH<=&H9F THEN JH=(SJH-&H71)*2+1 ELSE JH=(SJH-&HB1)*2+1
180 IF SJL>=&H80 THEN JL=SJL-&H1F-1 ELSE JL=SJL-&H1F
190 RETURN
200 *GREATER9EH
210 IF SJH<=&H9F THEN JH=(SJH-&H70)*2 ELSE JH=(SJH-&HB0)*2
220 JL=SJL-&H7E
230 RETURN
```

2.3 全角文字が使える部分

全角文字は，次のような部分に使うことができます．

(1) 文字列の内容
(2) 文字定数(" "で囲まれた文字列)
(3) REM 文の中の注釈
(4) DATA 文中の文字定数
(5) ファイル名
(6) INPUT(WAIT), LINE INPUT(WAIT)文のプロンプト文
(7) ファンクションキーの内容

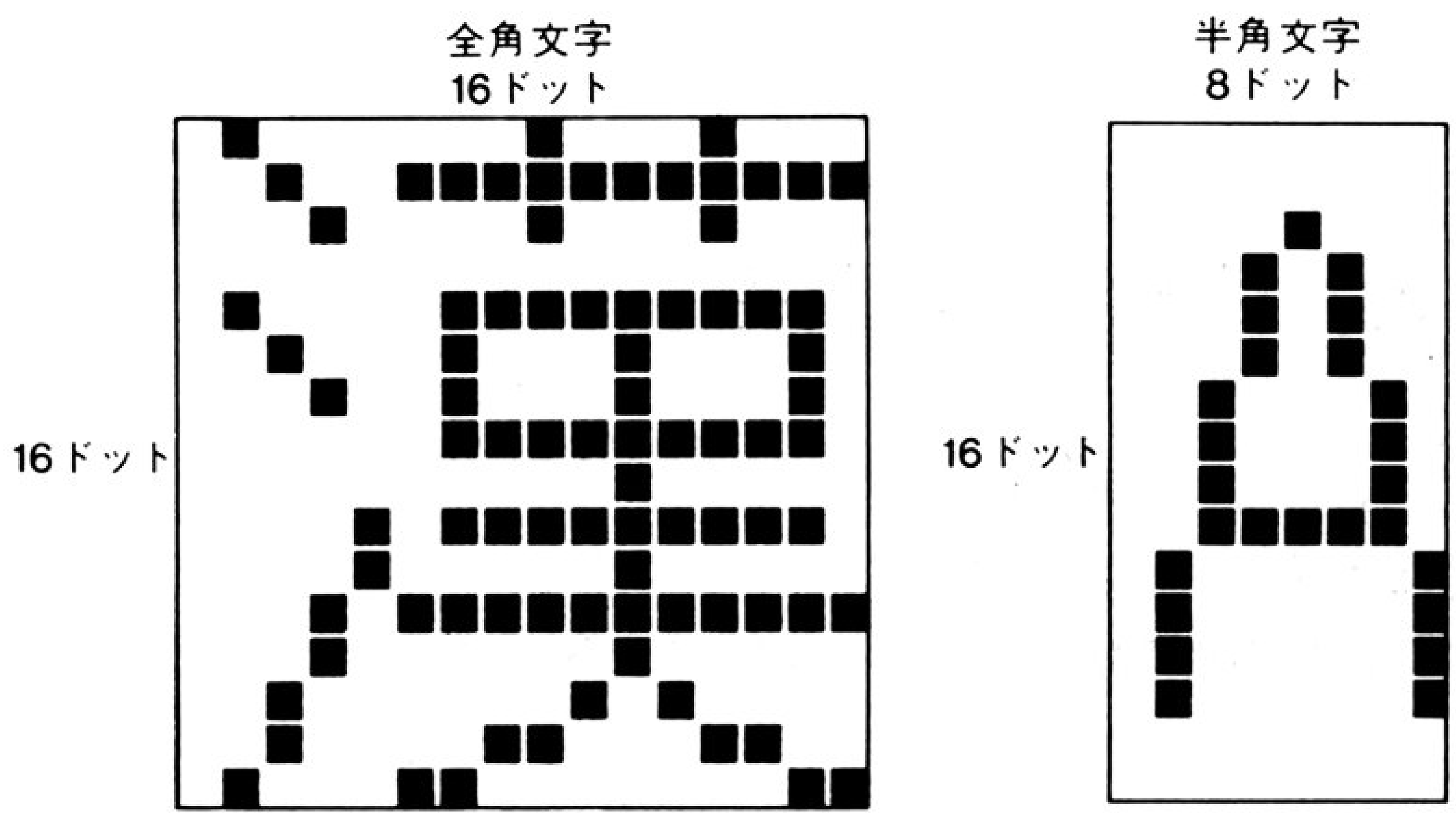
3. 画面構成

全角文字は，漢字などを表示する必要性から，右図のように半角文字とはドットパターンが異なります．

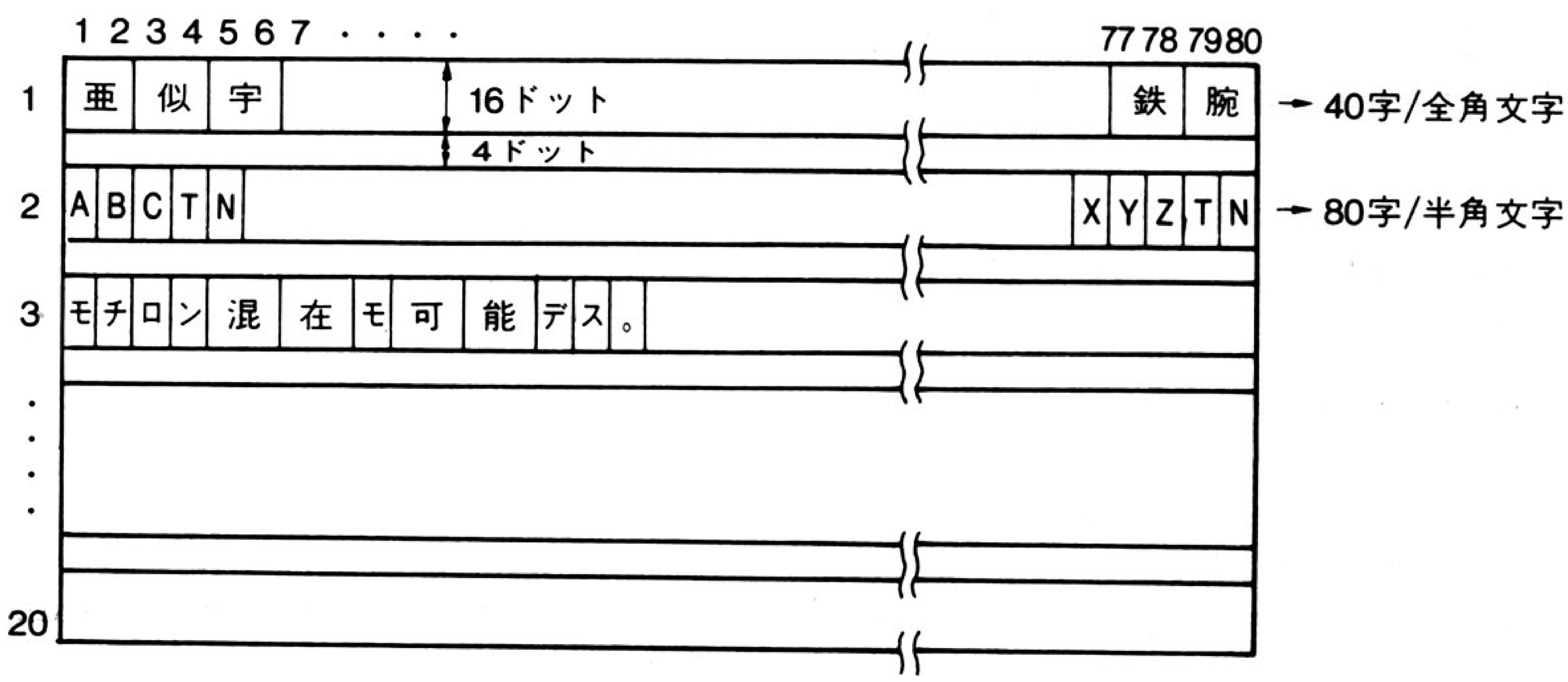
したがって，全角文字は半角文字2文字分のスペースを必要とします．つまり，1行をすべて全角文字で埋めた場合は，半角文字の半分の文字数しか表示できません．

もちろん全角文字と半角文字を混在させることもできます．

これを図に示すと，次のようになります．



例 WIDTH 80 , 20 : SCREEN 4の場合



この例の場合，1画面には40×20＝800文字が表示できることになります．

WIDTH文の第1パラメータ(1行中の文字数)は，半角文字で数えた値になりますから注意してください．

WIDTH文の設定と，そのときに表示できる全角文字とは，右表のような関係になります．

例 SCREEN 3の場合

WIDTH文の指定	表示できる全角文字の数
WIDTH 80, 12	40文字×12行＝480文字
WIDTH 80, 10*	40文字×10行＝400文字
WIDTH 40, 12	20文字×12行＝240文字
WIDTH 40, 10*	20文字×10行＝200文字

例 SCREEN 4の場合

WIDTH文の指定	表示できる全角文字の数
WIDTH 80, 25	40文字×25行＝1000文字
WIDTH 80, 20*	40文字×20行＝ 800文字
WIDTH 40, 25	20文字×25行＝ 500文字
WIDTH 40, 20*	20文字×20行＝ 400文字

* SCREEN文を実行すると，WIDTH文の第2パラメータ（行数/画面）は，自動的に10または20に設定されます．

4. 日本語処理用関数

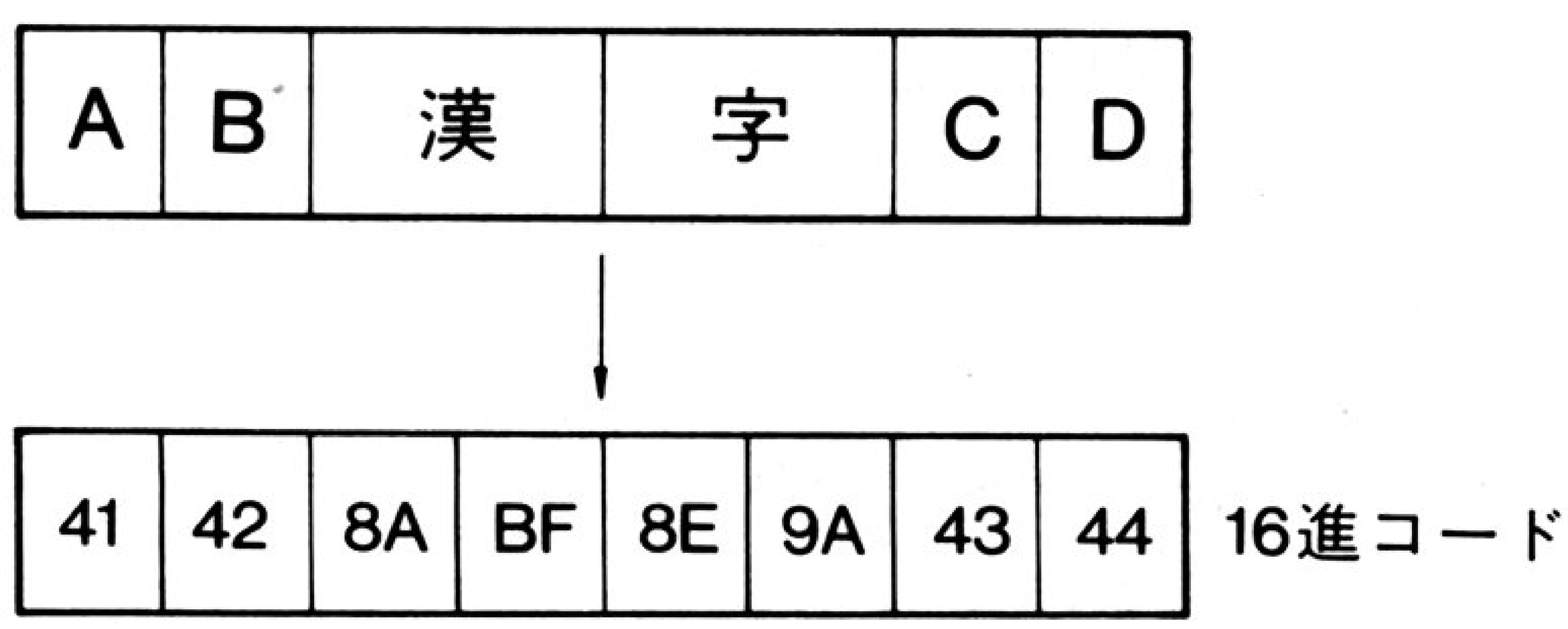
日本語処理用の関数として，KPOS, KLENなどがあります．

これらの関数に関する詳細は，BASICリファレンスマニュアルを参照してください．

BASIC中の文字および文字列を操作する関数，ステートメントは，すべて半角文字と全角文字を混在して扱えるようになっています．

このためINSTR関数，MID$関数，RIGHT$関数，LEFT$関数が期待どおりの動作をしないことがあります．これらの関数は，全角文字，半角文字を区別して扱わず，全角文字は半角文字が2個並んでいるものとして動作をするためです．

たとえば"AB漢字CD"という文字列は，右のように8バイトの半角文字とみなされます．



ここでは，MID$関数とINSTR関数を例にあげて，文字列の扱いに関する留意点を説明していきます．

1 MID$関数

(1)の例では，"AB海老"という文字列の3バイト目から2バイトの長さの文字列を切り取って表示させています．"海"は1文字で2バイトの長さがありますから，正しい結果が得られたわけです．

右のように指定すると，どうなるでしょう．

(2)の例を実際に実行してみると，おかしな結果になります．"AB海老"の4バイト目は，"海"の後ろ半分と，"老"の前半分を切り取ってしまったためです．このような文字列の切り分け方をすると，場合によって，おかしな文字が表示されたりするので注意が必要です．

全角文字を含む文字列を操作するときは，全角文字の半分だけを切り取ったりしないで，2バイト単位で(1文字分を)組みにして扱うようにしてください．

全角文字と半角文字との混在する文字列を扱う場合に，全角文字も半角文字と同じように1文字単位で扱いたい場合には，KLEN関数，KPOS関数と組み合わせてプログラムを組む必要があります．

LEFT$, RIGHT$, MID$ステートメントも同様の注意が必要です．

```
PRINT MID$("AB海老",3,2) ⏎ ——(1)
海
Ok
█
```

```
PRINT MID$("AB海老",4,2) ⏎ ——(2)
```

2 INSTR関数

右の例からわかるように，同じ文字"S"を探す場合に，うまくいく場合(1)と，うまくいかない場合(2)ができます．これは，BASICが全角文字をシフトJISコードで処理するために起こるものです．つまり，(2)の文字列中の"鬼"のシフトJISコードは8B53Hで，このうち下位バイトの53Hが"S"のキャラクタコードと同じため，BASICは，ここで一致した(発見した)とみなしてしまうからです．これは，調べたい文字列が半角文字1文字のときのみに起こり，半角文字が2文字以上，および全角

```
PRINT INSTR("日本語BASIC","S") ⏎ ——(1)
 9
Ok
█
```

```
PRINT INSTR("鬼が島BASIC","S") ⏎ ——(2)
 2
Ok
█
```

文字のときには生じません．この問題を解決するサブルーチンを含むプログラムリストを右に掲げます．

```
100 'INSTR 補正プログラム
110 '
120 X$="鬼が島BASIC"
130 Y$="S"
140 N=1
150 '
160 GOSUB *ADVANCED.INSTR
170 PRINT Y$;" は、";X$;" の";NO;"バイトめにあります。
180 END
190 '
200 *ADVANCED.INSTR
210 IF LEN(Y$)=1 AND ASC(Y$)<256 THEN 250
220 ' Y$が、2バイト以上のとき
230 NO=INSTR(N,X$,Y$)
240 GOTO 330
250 ' Y$が、半角文字1文字のとき
260 Y=ASC(Y$):I=N
270 C=KPOS(X$,I)
280 IF C=0 THEN 330
290 W$=MID$(X$,C,1)
300 IF ASC(W$)=Y THEN NO=I
310 IF NO=0 THEN I=I+1:GOTO 270
320 NO=KPOS(X$,NO)
330 RETURN
```

このプログラムは，半角文字が1文字のときだけ別のサブルーチンで判断しているので，正しい結果が得られるわけです．

このプログラムでは，X$に調べる文字列を代入し，Y$に調べたい文字列を代入します．そして，何バイト目から調べるかをNに代入します(行番号120～140)．それから，200行から始まるサブルーチンを実行します．

サブルーチン中では，Y$が1文字で半角文字の場合と，そうでない場合の区別をします(行番号210)．そして，Y$が1文字で半角文字の場合は250行へ実行が移り，ここで正しい判断をします．また，そうでない場合は，INSTR関数で処理できるので220行へと実行が移ります．どちらもNOという変数に答えを代入して，330行のRETURN文でサブルーチンを終わります．その後，170行に帰りNOを表示してプログラムを終了します．

これにより，INSTR関数は正確に動作できるようになります．したがって，INSTRで半角文字1文字の判断をするときは，このプログラムの250行から310行を参考にして，プログラムを作成してください．

さて，INSTR関数では，その文字(調べる文字列)が左から何バイト目にあるかを返すものですが，"何文字目にあるか"を得たい場合があります．特に全角文字を使用していると，INSTR関数では，バイト数＝文字数が成立しないので，文字列操作がやっかいなものになります．そこで，"拡張INSTR関数"とでも言うプログラム(サブルーチン)を掲げておきます．

```
100 ' 拡張INSTR関数プログラム
110 X$="鬼が島BASIC"
120 Y$="S"
130 N=1
140 '
150 GOSUB *ENHANCED.INSTR
160 PRINT Y$;" は ";X$;" の";NO;"文字めにあります。"
170 END
180 *ENHANCED.INSTR
190 IF LEN(Y$)=1 AND ASC(Y$)<256 THEN 270
200 '
210 'Y$が、2バイト以上のとき
220 NO=INSTR(N,X$,Y$):IF NO=0 THEN 340
230 C$=LEFT$(X$,NO-1)
240 C=LEN(C$)-KLEM(C$):NO=NO-C
250 GOTO 340
260 '
270 'Y$が、半角文字1文字のとき
280 Y=ASC(Y$):I=N
290 C=KPOS(X$,I)
300 IF C=0 THEN 340
310 W$=MID$(X$,C,1)
320 IF ASC(W$)=Y THEN NO=I
330 IF NO=0 THEN I=I+1:GOTO 290
340 RETURN
```

これにより，NOという変数に文字数(半角文字と全角文字混在のキャラクタ数)を返すことができます．プログラムの内容については，ほとんどINSTR関数のプログラムと同じなので，そちらを参照してください．

5. 日本語プリンタへの印字

N88-日本語BASICは，漢字などの全角文字を出力できる日本語プリンタをサポートしています．LPRINT 文や LLIST 文を使って，全角文字を含む文字列やプログラムを日本語プリンタに印字することができます．また，COPY 文や，CTRL＋COPY で，プリンタの文字フォントを使った画面コピーも可能となっています．

ここでは，主なプリンタのうち，N88-日本語BASICに適したプリンタ，使用するときに注意が必要なプリンタなどについて解説します．

(1) N88-日本語BASICに適したプリンタ

右のプリンタは，N88-日本語BASICのすべての機能がサポートされます．

24ドット日本語プリンタ	PC-PR101L
	PC-PR101T
	PC-PR201H
	PC-PR201T
	PC-PR201CL
	PC-PR201HC
	PC-PR406

(2) 注意が必要なプリンタ

半角文字と全角文字の比が１：２にならないプリンタ

右のプリンタに全角文字を含む文字列を印字した際，半角文字と全角文字の比が，１：２になっていませんので，桁が揃わない場合があります．

24ドット日本語プリンタ	PC-PR104
	PC-PR405
18ドット日本語プリンタ	PC-PR202K
16ドット日本語プリンタ	PC-PR403

これを，補正するためには，システムディスクに入っているユーティリティを使用してください．

システムディスクがドライブ１に入っていることを確認した上で，

```
run "setpdt . j88"
```

を実行します．

外字が使用できないプリンタ

外字機能を持つ日本語プリンタであっても，ドット数の違いから外字機能を使用することができないプリンタがあります．

18ドット日本語プリンタ	PC-PR202K
16ドット日本語プリンタ	PC-PR403

(3) 全角文字が印字できないプリンタ

右のプリンタは，全角文字を印字することができません．

PC-8024*
PC-8027*
PC-2021
PC-6023
PC-8826
PC-PR103
PC-PR401
PC-PR402

* オプションの漢字ROMを付けると，全角文字が印字できます．

6．注意事項

(1) 文字列定数は最大255桁ですが，全角文字を含む場合，全角文字の1文字が半角文字2文字に対応しているため，実際の有効文字数はそれ以下となります．

すべて全角文字からなる文字定数の最大文字数は127文字です．

(2) ランダムファイルに日本語文字列を含んだデータを出力する場合，全角文字1文字が英数カナ文字2文字に対応することを考えて，フィールド長を決定する必要があります．

例

```
10 a$="日本語"
20 lset fa$=a$
30 put #1,i
```

この場合，フィールドfa$のフィールド長は最低6文字必要となります．

```
field #1,10 as fa$,………
```

(3) N_{88}-日本語BASICの日本語モードでCOPY 1を行った場合，ファンクションキーの内容はコピーされません．

第4章 いろいろな音を出す

PC-8801mkIIMRには
シンセサイザIC(OPN)が内蔵されており,
強力なサウンド機能を持っています.
6重奏や効果音を出すことができ,
さらに自分で音色を定義することもできます.

ここでは，OPNを使って簡単に音を出す方法を紹介しましょう．音を出すには，拡張命令であるCMD PLAY文を使います．

N_{88}-BASIC V2またはN_{88}-日本語BASICを起動させたあと，右のように入力してください．これで拡張命令が使えるようになりました．

```
NEW CMD⏎
```

では音を出してみましょう．右のようにします．

"ドレミファソ"と音が出ましたか？もし音が出ないときは，アンプのボリューム，あるいはケーブルの接続などを確かめてください．

```
CMD PLAY "V15O4L4CDEFG"⏎
```

今出た音は，デフォルトボイスのハープシコードの音です．このほかにも約50種類の違った音色が用意されています．それでは，これらの中からいくつかの音を出してみましょう．音色の変更は，CMD PLAY文で"@"のパラメータとして音色番号を与えます．音色番号と音色の対応は，BASICリファレンスマニュアルのCMD PLAYの音色番号表を参照してください．

まず初めに，音色番号8のオーボエの音を選びます．右のようにします．

```
CMD PLAY "@8CDEFG"⏎
```

先ほど実行した命令では，ボリューム(V)，オクターブ(O)，音の長さ(L)の設定をしたのに，今度は設定しませんでした．これらの命令は，その前に実行したときの設定値を覚えているからです．しかし，音が出ているときに STOP を押して途中で音を止めたり，CMD STOPM命令を実行するとこれらの値はデフォルト値になりますので，その時はもう一度設定してください．

それでは，また別の音色に変えて音を出してみましょう．右のようにします．

これはエレクトリックピアノの音です．

```
CMD PLAY "@26CDEFG"↵
```

これはパイプオルガンの音です．今までは単なる楽器の音でしたが，ここで少し変わったものを紹介しましょう．

```
CMD PLAY "@34CDEFG"↵
```

今度は音を少し長くしてみます．右のようにします．

```
CMD PLAY "@49L1CDEFG"↵
```

ブラスのような感じの音ですね．同じ音色で今度は音を短くして出します．右のようにします．

```
CMD PLAY "L64CR1DR1ER1FR1G"↵
```

どうですか？ ベルの音になったと思いませんか？ このほかにも，変わったものとして効果音のような音色があります．右のようにします．

```
CMD PLAY "@54L105C&C&C&C&C"↵
```

救急車の音のようですね．今はタイ（&）を使って音を長くしてみました．これは，音が短いと救急車のサイレンの効果が表れないために音を長くしたのです．このように効果音は，音の出し方，つまり音の長さや音の高さによっては，その効果が表れなかったり，また違った効果が得られます．

ほかにもまだ変わった音色がありますので，いろいろな音を出して楽しんでください．そして自分で音色が作りたくなったら，資料3　音の原理を読んでください．

第5章 グラフィック命令を使う

N88-BASIC/N88-日本語BASICの グラフィック命令には，LINE，CIRCLE，PAINT，GET@，PUT@などさまざまな命令があります．ここでは，代表的なグラフィック命令として，VIEW，WINDOWそしてCMD PALを使ったプログラムを紹介します．

1．ウィンドウとビューポートの使い方

ここでは，ウィンドウとビューポートの概念を具体的に目で見て理解していただくために，全く同一のグラフを表示するサブルーチンを題材に説明します．これは，4象限の直交座標に"Y＝X"，"Y＝−2＊X＋15"，"Y＝X＊X/50−30"の関数をグラフ表示するものです．

VIEW sample programは，ウィンドウを固定しておき(160行)，ビューポートの大きさを変化させながら，先ほどのグラフ表示サブルーチン(270行以降)を呼んでいます．実行結果は，グラフそのものの形状および情報はそのままで，表示される領域の大きさだけが変化していきます．

```
100 '--  VIEW sample program  --
110 '
120  DEFINT A-Z
130  WIDTH 80,25:CONSOLE 0,25,0,1
140  SCREEN 0,0:COLOR 7,0:CLS 3
150  LOCATE 29,0:PRINT "VIEW demonstration"
160  WINDOW(-110,-110)-(110,110)
170 '
180  FOR I=1 TO 5
190    VIEW(320-310/I,100-90/I)-(320+310/I,
       100+90/I),0,3
200    GOSUB *GRAPH
210    FOR J=0 TO 2000:NEXT J
220  NEXT I
230  SCREEN 0
240  END
250 '-- graph display routine --
260 *GRAPH
270  LINE(-100,0)-(100,0),7
280  LINE(0,-100)-(0,100),7
290  FOR X=-80 TO 80
300    PSET(X,-X),1              ' Y=X
310    PSET(X,-(-2*X+30)),2      ' Y=-2*X+15
320    PSET(X,-(X*X/50-30)),4    ' Y=X*X/50-30
330  NEXT X
340  RETURN
```

WINDOW sample program は VIEW sample programの150，160，190行を変更し，215行を1行追加しただけのものです．VIEWの場合とは逆にビューポートを固定しておき(160行)，ウィンドウを変化させながらグラフ表示サブルーチンを呼んでいます．その結果，画面上の表示領域の大きさは常に同じですが，表示されるグラフはだんだん拡大されたものになっていきます．

このようにウィンドウとビューポートを適当に変化させることによって，同じ表示のサブルーチンであっても，画面上の任意

```
100 '--  WINDOW sample program  --
110 '
120  DEFINT A-Z
130  WIDTH 80,25:CONSOLE 0,25,0,1
140  SCREEN 0,0:COLOR 7,0:CLS 3
150  LOCATE 29,0:PRINT "WINDOW demonstration"
160  VIEW(100,8)-(500,198),,3
170 '
180  FOR I=1 TO 5
190    WINDOW(-200/I,-200/I)-(200/I,200/I)
200    GOSUB *GRAPH
210    FOR J=0 TO 2000:NEXT J
215    CLS 2
220  NEXT I
230  SCREEN 0
240  END
250 '-- graph display routine --
260 *GRAPH
270  LINE(-100,0)-(100,0),7
280  LINE(0,-100)-(0,100),7
```

の領域に拡大，縮小した図形が表示できますから，グラフのアレンジ，画面上でのレイアウトなど容易に行うことができます．

```
290  FOR X=-80 TO 80
300    PSET(X,-X),1                 ' Y=X
310    PSET(X,-(-2*X+30)),2         ' Y=-2*X+15
320    PSET(X,-(X*X/50-30)),4       ' Y=X*X/50-30
330  NEXT X
340  RETURN
```

2. アナログパレットの使い方

N₈₈-BASIC V2では，拡張命令を追加することによって，512色の中から8色を選んで同一画面上に表示することができます．その選んだ色をパレットに設定するのがCMD PAL文です．

次のサンプルプログラムは，8色のカラーを画面に表示し，その中の1つのパレットを指定し(270行)，青の輝度，赤の輝度，緑の輝度を指定(300行から410行)することによって，指定したパレットの色を変えます．パレットナンバーを指定するところで"−1"を入力すると，パレットを初期状態に戻してから終了します．パレットを変更した場合，必ずこのようにパレットを初期状態に戻すよう心がけてください．

また，このプログラムを実行する場合は，本体にアナログRGBディスプレイを接続することが必要です．

```
100 '--  ANALOG PALETTE sample program  --
110 '
120 WIDTH 80,25:CONSOLE 0,25,0,1:SCREEN 0,0
130 NEW CMD:CLS 3:CMD PAL
140 LOCATE 20,1
150 PRINT "ANALOG PALETTE sample program"
160 FOR X=40 TO 544 STEP 72
170   LINE(X,50)-(X+71,138),INT(X/70),BF
180 NEXT X
190 FOR I=0 TO 7
200   LOCATE 8+CX,5:PRINT I
210   CX=CX+9
220 NEXT I
230 '
240 *LOOP
250  DEC=0
260  LOCATE 2,19
270  INPUT "PALETTE NO. (0-7 or -1=end) ";A
280   IF A=-1 THEN *EXIT
290   IF A<-1 OR A>7 THEN LOCATE 32,19:
                          PRINT SPACE$(48):
                          GOTO 260
300  LOCATE 45,19
310  INPUT "GREEN INTENSITY ( 0 - 7 ) ";B
320   B$=MID$(STR$(B),2)
330   IF B<0 OR B>7 THEN LOCATE 72,19:
                         PRINT SPACE$(8):
                         GOTO 300
340  LOCATE 2,21
350  INPUT "RED INTENSITY ( 0 - 7 ) ";C
360   C$=MID$(STR$(C),2)
370   IF C<0 OR C>7 THEN LOCATE 28,21:
                         PRINT SPACE$(52):
                         GOTO 340
380  LOCATE 45,21
390  INPUT "BULE INTENSITY ( 0 - 7 ) ";D
400   D$=MID$(STR$(D),2)
410   IF D<0 OR D>7 THEN LOCATE 71,21:
                         PRINT SPACE$(9):
                         GOTO 380
420  LEVEL$=B$+C$+D$
430  FOR I=1 TO 3
440    NUM=VAL(LEFT$(LEVEL$,1))
450    LEVEL$=MID$(LEVEL$,2)
460    DEC=8*DEC+NUM
470  NEXT
480  CMD PAL A,DEC
490  CONSOLE 19,6:CLS
500 GOTO *LOOP
510 '
520 *EXIT
530  CMD PAL
540  CONSOLE 0,25
550 END
```

第6章 データファイルを作る（フロッピーディスクを使って）

データファイルには，シーケンシャルファイルとランダムファイルの2種類があります．カセットを対象とする場合は，ランダムファイルを作ることはできませんが，シーケンシャルファイルは作ることができます．ここでは，フロッピーディスクドライブを使用した場合とカセットテープレコーダを使用した場合に分けて説明します．

1．シーケンシャルファイル

シーケンシャルファイルは，ランダムファイルよりも簡単に作ることができますが，データをアクセスするときに，融通性とスピードの点でランダムファイルより制限を受けます．これは，シーケンシャルファイルがデータを順番に記録し，読み出すときも最初から順番に読まなければならないからです．それに対しランダムファイルは，任意のレコード番号を指定することによって，それに対応するデータを取り出すことができます．

シーケンシャルファイルをアクセスする際に使用されるステートメントと関数を以下に示します．

OPEN, CLOSE, INPUT#, LINE INPUT#, PRINT#, PRINT# USING, WRITE#, EOF, LOF, LOC, INPUT$(X, [#]Y)

シーケンシャルファイルを作り，その中のデータをアクセスするために必要な手順は次のようになります．

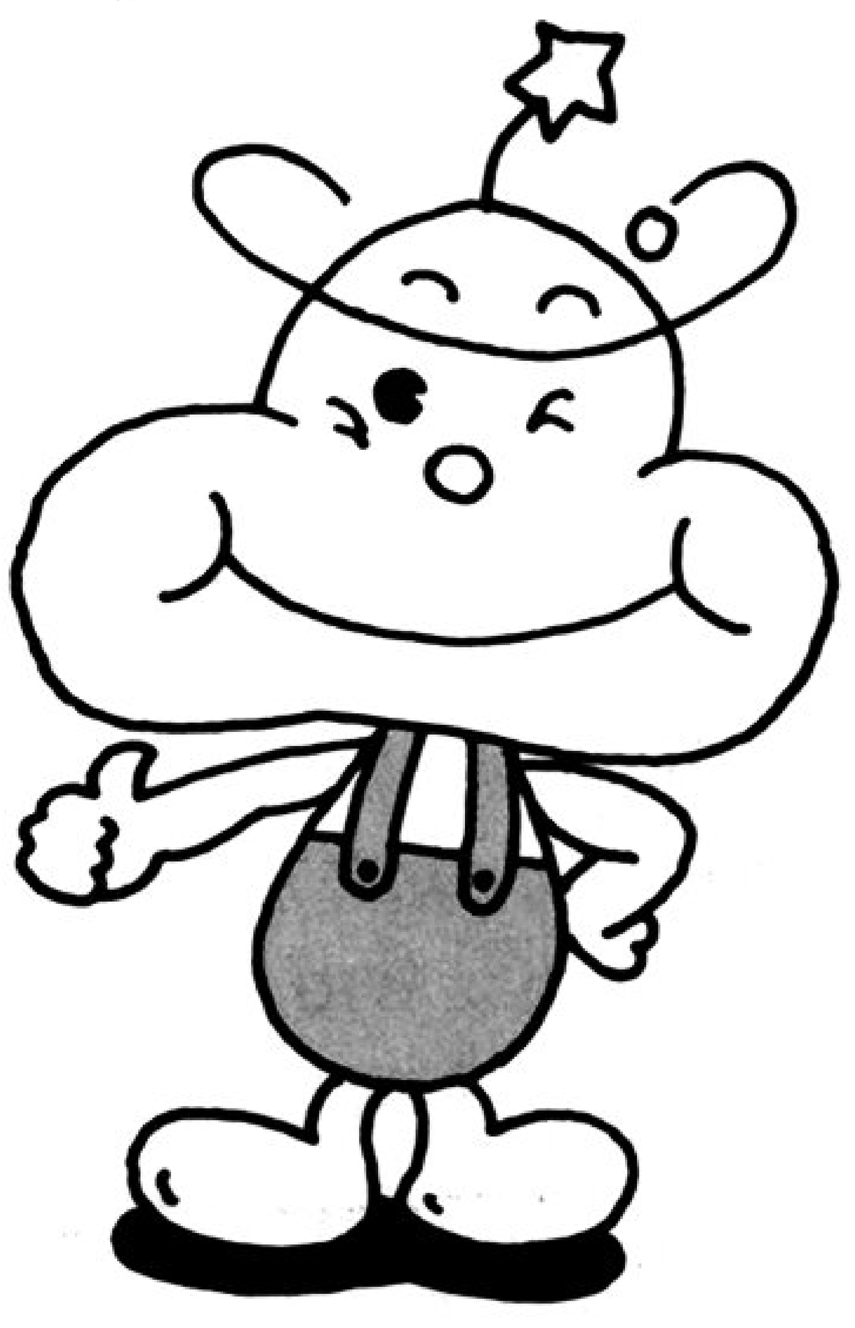
シーケンシャルファイルの作成

(1) 書き込みのためのファイルをオープンします．

```
OPEN "TEST" FOR OUTPUT AS #1
```

(2) WRITE#, PRINT#等を使って，ファイルにデータを書き込みます．

```
WRITE #1, N$; P
```



(3) 必要なだけ(2)の作業を繰り返した後，ファイルをクローズして終了します．

　　CLOSE #1

例として，キーボードから生徒のテストの点数を入力して，シーケンシャルファイル"TEST"を作るプログラムを示します（プログラム1）．

```
100 ' ﾌﾟﾛｸﾞﾗﾑ 1
110  OPEN "TEST" FOR OUTPUT AS #1
120 *LOOP
130  INPUT "ﾅﾏｴ ";N$
140  IF N$="end" THEN *FIN
150  INPUT "ｶﾓｸ ";K$
160  INPUT "ﾄｸﾃﾝ";P
170  WRITE #1,N$;K$;P
180  PRINT
190 GOTO *LOOP
200 *FIN
210  CLOSE #1:END


run
ﾅﾏｴ ? ﾎﾝﾀﾞ ﾀｸﾔ
ｶﾓｸ ? ﾘｶ
ﾄｸﾃﾝ? 75

ﾅﾏｴ ? ｽｽﾞｷ ﾗﾝ
ｶﾓｸ ? ﾘｶ
ﾄｸﾃﾝ? 48

ﾅﾏｴ ? ﾔﾏﾊ ﾘｻ
ｶﾓｸ ? ﾘｶ
ﾄｸﾃﾝ? 47

ﾅﾏｴ ? end
 OK
```

シーケンシャルファイルを作成するプログラムにおいて，PRINT# USINGを使用することによって，書式制御したデータを書き込むこともできます．たとえば，

　　PRINT #1,USING "####.##,";
　　A,B,C,D

を使うと，特に区切り記号を付けることなく，ディスクに数値データを書き込むことができます．この区切り記号とは，シーケンシャルファイル中のデータの読み込みにINPUT#を使用するとき，数値データの場合はコンマ，キャリッジリターン，ラインフィード，空白などを区切りとしてデータを読み込むためのものです．LINE INPUT#を使用する場合は，キャリッジリターンのみを区切り記号とします．また，PRINT#の代わりにWRITE#を使用しますと，区切り記号を強制的に出力するため，ディスクの使用領域を節約できます．

LOC関数をシーケンシャルファイルについて使用する場合は，そのファイルが開かれて以降，アクセスされた総セクタ数を値とします．

シーケンシャルファイルからの読み込み

(1) 読み込みのためのファイルをオープンします.

OPEN "TEST" FOR INPUT AS #1

(2) INPUT#, LINE INPUT#等を使って，ファイルからデータを読み込みます.

INPUT #1, N$, P

(3) 必要なだけ(2)の作業を繰り返した後，ファイルをクローズして終了します.

CLOSE #1

次にプログラム2として，プログラム1で作ったファイル"TEST"をアクセスし，50点以下の生徒を全部表示する例を右に示します.

```
100 ' ﾌﾟﾛｸﾞﾗﾑ 2
110  OPEN "TEST" FOR INPUT AS #1
120 *LOOP
130  INPUT #1,N$,K$,P
140  IF P>=50 THEN *LOOP
150  PRINT N$,K$,P
160 GOTO *LOOP

run
ｽｽﾞｷ ﾗﾝ        ﾘｶ              48
ﾔﾏﾊ ﾘｻ         ﾘｶ              47
Input past end in 130
Ok
```

プログラム2では，ファイル中のデータを最初から順番に読み込んでいきます．そして，ファイルの最後でもう読み込むデータがなくなると，130行で"Input past end"エラーが起こります．これを避けるためには，125行に右の1行を挿入します.

```
125  IF EOF(1) THEN *FIN
```

ファイルの終わりをチェックし，右の3行のように処理します.

```
170 *FIN
180  CLOSE #1
190 END
```

シーケンシャルファイルへのデータの追加

フロッピーディスク上にシーケンシャルファイルがすでに作られていて，その後にデータを追加したい場合には，次のような手順で行います.

(1) 追加するためのファイルをオープンします.

OPEN "TEST" FOR APPEND AS #1

(2) WRITE#, PRINT#等を使って，ファイルに追加したいデータを書き込みます.

WRITE #1, N$; P

(3) 必要なだけ(2)の作業を繰り返した後，ファイルをクローズして終了します．

CLOSE #1

例として，プログラム1で作ったファイル"TEST"にデータを追加する例をプログラム3として示します．

```
100 ' ﾌﾟﾛｸﾞﾗﾑ 3
110  OPEN "TEST" FOR APPEND AS #1
120 *LOOP
130  INPUT "ﾅﾏｴ ";N$
140  IF N$="end" THEN *FIN
150  INPUT "ｶﾓｸ ";K$
160  INPUT "ﾄｸﾃﾝ";P
170  WRITE #1,N$;K$;P
180  PRINT
190 GOTO *LOOP
200 *FIN
210  CLOSE #1:END
```

```
run
ﾅﾏｴ ? ﾎﾝﾀﾞ ﾀｸﾔ
ｶﾓｸ ? ｺｸｺﾞ
ﾄｸﾃﾝ? 46

ﾅﾏｴ ? ｽｽﾞｷ ﾗﾝ
ｶﾓｸ ? ｺｸｺﾞ
ﾄｸﾃﾝ? 74

ﾅﾏｴ ? ﾔﾏﾊ ﾘｻ
ｶﾓｸ ? ｺｸｺﾞ
ﾄｸﾃﾝ? 80

ﾅﾏｴ ? end
Ok
```

このプログラムは，プログラム1の110行のOUTPUTをAPPENDに代えただけです．したがって，このプログラムでデータを追加したファイルに対して，プログラム2を実行してチェックすることができます．

```
ｽｽﾞｷ ﾗﾝ          ﾘｶ                     48
ﾔﾏﾊ ﾘｻ           ﾘｶ                     47
ﾎﾝﾀﾞ ﾀｸﾔ         ｺｸｺﾞ                   46
Input past end in 130
Ok
```

シーケンシャルファイルのデータの修正

シーケンシャルファイルでは，そのファイルのデータを修正することができません．したがって，現在のファイルから読み込んで修正したデータを新しい別のシーケンシャルファイルへ書き込んでいく作業を行わなければなりません．右にプログラム4として，プログラム1や3で作ったファイルの修正を行う例を示します．

プログラム4では，ファイル"TEST"から読み込んだデータに修正がない場合はそのまま新しいファイル"test"へ書き込みます．もし修正がある場合は，260〜280行で新しいデータを入力し，"test"へ書き込みます．削除する場合には，書き込まずに次のデータを"TEST"から読み込みます．修正が終了した場合には，"TEST"から読み

```
100 ' ﾌﾟﾛｸﾞﾗﾑ 4
110  OPEN "TEST" FOR INPUT AS #1
120  OPEN "test" FOR OUTPUT AS #2
130 *LOOP
140  IF EOF(1) THEN *FIN
150  INPUT #1,N$,K$,P
160  IF FLG=1 THEN *WR
170  PRINT N$,K$,P
180 *QUESTION
190  PRINT "ﾍﾝｺｳ ｶﾞ ｱﾘﾏｽｶ ";
200  INPUT "(y or n or d or e )";A$
210  A$=LEFT$(A$,1)
220  IF A$="e" THEN FLG=1:GOTO *WR
230  IF A$="d" THEN *LOOP
240  IF A$="n" THEN PRINT:GOTO *WR
250  IF A$<>"y" THEN *QUESTION
260  INPUT "ﾅﾏｴ ";N$
270  INPUT "ｶﾓｸ ";K$
280  INPUT "ﾄｸﾃﾝ";P
290  PRINT
300 *WR
310  WRITE #2,N$;K$;P
320 GOTO *LOOP
330 *FIN
340  CLOSE #1
350  CLOSE #2
360  KILL "TEST"
```

込んだデータをそのまま"test"へ書き込む作業を"TEST"のファイルエンドまで繰り返します.

最後に，古いファイル"TEST"を消去し，"test"を"TEST"に名前を変更して終了します.

```
370  NAME "test" AS "TEST"
380 END
```

2. ランダムファイル

ランダムファイルを作るには，シーケンシャルファイルの場合より多くの手順が必要となります.

しかし，ランダムファイルではデータをコンパクトに記録するため，ファイルの占める領域を小さくすることができます．また，レコード単位で任意にデータのアクセスができるため，シーケンシャルファイルの場合のように空読みする必要がありません．さらに，データの修正を自由に行えるという利点もあります.

ランダムファイルをアクセスする際に使用できるステートメント，および関数は次のとおりです.

OPEN, CLOSE, FIELD, LSET, RSET, GET, PUT, CVI/CVS/CVD, MKI$/MKS$/MKD$, LOC, LOF

ランダムファイルを作ったりアクセスしたりするには，次の手順が必要となります.

(1) ランダムファイルをオープンします.

```
OPEN "TESTR" AS #1
```

(2) ランダムファイルバッファに文字変数名を与えます.

```
FIELD #1,20 AS N$,20 AS K$,2 AS P$
```

(3) データを書き込む場合は，その前にバッファへデータを転送しておきます．データが文字列の場合はLSET, RSETを使います.

```
LSET N$=A$
```

また，数値である場合には，一度MKI$, MKS$, MKD$を使って，それぞれ整数値，単精度実数値，倍精度実数値を2バイト，4バイト，8バイトの文字に変換してから



バッファへ転送します.

LSET P$=MKI$(P)

(4) データの書き込み，読み込みを行います.

PUT #1,NUM%

GET #1,NUM%

GET#によって読み込んだデータが数値である場合には，CVI, CVS, CVDを使って文字列から数値へ変換します.

P=CVI(P$)

(5) ランダムファイルをクローズして，作業を終了します.

CLOSE #1

右のプログラム5は生徒番号によって，ランダムに読み書きするものです. 新しくファイルを作る場合は， RUN実行後に"ファイルメイ"を入力したら， 次に必ず"ファイル　ノ　イニシャライズ"を実行してください.

```
100 ' ﾌﾟﾛｸﾞﾗﾑ 5
110  CLS:INPUT "ﾌｧｲﾙ ﾒｲ   ";F$
120  OPEN F$ AS #1
130  FIELD #1,1 AS ID$,20 AS N$,20 AS K$,2 AS P$
140 *FUNC
150  CLS
160  PRINT "   1:ﾃﾞｰﾀ ｻｸｾｲ,ﾍﾝｺｳ"
170  PRINT "   2:ﾃﾞｰﾀ ｻﾝｼｮｳ"
180  PRINT "   3:ﾃﾝｽｳ ﾆ ﾖﾙ ｹﾝｻｸ"
190  PRINT "   4:ｼｭｳﾘｮｳ"
200  PRINT "999:ﾌｧｲﾙ ﾉ ｲﾆｼｬﾗｲｽﾞ"
210  INPUT "   ｴﾗﾝﾃﾞｸﾀﾞｻｲ";F
220   IF F=999 THEN *FINI
230  ON F GOTO *HENKO,*SANSHO,*KENSAKU,*FIN
240 GOTO *FUNC
250 *SANSHO
260  SFLG=-1
270 GOTO *HKLOOP
280 *HENKO
290  SFLG=0
300 *HKLOOP
310  GET #1,1
320  MAX=CVI(P$)
330  PRINT "ｾｲﾄ ﾊﾞﾝｺﾞｳ (1-";USING "###";MAX-1;
340  INPUT " or 0:end)";NUM%
350   IF NUM%=0 THEN *FUNC
360  NUM%=NUM%+1
370  GET #1,NUM%
380   IF ID$=CHR$(255) AND SFLG=0 THEN *QUESTION
390   IF ID$=CHR$(255) AND SFLG THEN *HKLOOP
400  P%=CVI(P$)
410  PRINT
420  PRINT "ﾅﾏｴ :";N$
430  PRINT "ｶﾓｸ :";K$
440  PRINT "ﾄｸﾃﾝ :";P%
450  PRINT
460   IF SFLG THEN *HKLOOP
470 *QUESTION
480  INPUT "ﾅﾏｴ ";A$
490   IF A$<>"" THEN LSET N$=A$
500  INPUT "ｶﾓｸ ";A$
510   IF A$<>"" THEN LSET K$=A$
520  INPUT "ﾄｸﾃﾝ ";A$
530   IF A$<>"" THEN LSET P$=MKI$(VAL(A$))
540  PRINT
550  LSET ID$="U"
560  PUT #1,NUM%
570 GOTO *HENKO
580 *KENSAKU
590  GET#1,1
600  MAX=CVI(P$)
610  INPUT "ﾅﾝﾃﾝｲｶ ﾃﾞ ｹﾝｻｸ ｼﾏｽｶ ";AKA
620  PRINT
630  FOR NUM%=2 TO MAX
640   GET #1,NUM%
650    IF ID$=CHR$(255) THEN *KLOOP
660    IF CVI(P$)>AKA THEN *KLOOP
```

```
670   PRINT USING "### ";NUM%-1;
680  PRINT N$;K$;USING " ####";CVI(P$)
690 *KLOOP
700  NEXT NUM%
710  PRINT
720  PRINT "ｹﾝｻｸ ｼｭｳﾘｮｳ. ｷｰ ｦ ﾀﾀｲﾃ ｸﾀﾞｻｲ"
730 *LOOP
740 A$=INKEY$:IF A$="" THEN *LOOP
750 GOTO *FUNC
760 *FINI
770  INPUT "ｻｲﾀﾞｲ ﾉ ｾｲﾄﾊﾞﾝｺﾞｳ ﾊ ";MAX
780  INPUT "ﾌｧｲﾙ ｦ ｲﾆｼｬﾗｲｽﾞ ｼﾏｽ (yes or no) ";A$
790   IF A$<>"yes" THEN *FUNC
800  FOR NUM%=1 TO MAX+1
810   LSET ID$=CHR$(255)
820   PUT #1,NUM%
830  NEXT NUM%
840  LSET P$=MKI$(MAX+1)
850  PUT #1,1
860 GOTO *FUNC
870 *FIN
880  CLOSE #1
890 END
```

第7章 データファイルを作る（カセットテープを使って）

PC-8801mkIIMRには，カセットテープ用のインタフェースが装備されていません．この章で説明する操作をするためには，オプションのCMTインタフェースボードとデータレコーダが必要になります．カセットテープレコーダを使用してデータの書き込み，読み込みを行う場合は，リモート端子(黒)を接続します．

リモート端子を接続しますと，テープレコーダのON/OFFは，PC-8801mkIIMRがコントロールします．ですから，PLAYボタンを押していても，PC-8801mkIIMRでカセットテープを使用する命令が実行されなければカセットテープは回りません．この機能によって，必要な間だけカセットテープを回すことができ，カセットテープのむだづかいが防げ，また手間もかかりません．

リモート端子を接続しなかった場合は，テープレコーダのスタート，ストップはすべて自分でやる必要があります．ですから，カセットを使用する命令を実行する前にPLAYボタンを押し，終わるとSTOPボタンを押すという操作が必要になります．

以後，リモート端子が接続されているとして説明します．

1．データの書き込み

(1) キーボードから右のプログラムを入力してください．

```
100 INPUT A,B,C,D
110 PRINT "Set tape,then push PLAY and REC"
120 INPUT "Ok";A$
130 OPEN "cas:suchi" FOR OUTPUT AS #1
140 PRINT #1,A,B
150 PRINT #1,C,D
160 CLOSE #1
170 END
```

このプログラムを簡単に説明しましょう．

(a) キーボードから数値を入力します(100行)．

(b) カセットテープをセットし，録音ボタンとPLAYボタンを押して ⏎ を入力します(110，120行)．

(c) ファイルのオープンを行います．OPEN文が実行されると，カセットテープへのデータを出力するための窓口(バッファ)が割り当てられ，データ

を書き込むための準備が整います(130行).

(d) PRINT#文を用いてデータを書き込みます(140, 150行).

(e) データの書き込みが終わったら, CLOSE文でファイルを閉じておきます(160行).

(2) このプログラムを実行させます.

run

(3) すると, 画面に" ? "を表示します. これは, 行番号100のINPUT文がキーボードからの入力を要求しているのです.

1,2,3,4

と入力します.

(4) すると, 今度は次のメッセージが表示されます.

Set tape, then push PLAY and REC
Ok?

(5) そこで, データを入力してもよいカセットテープをテープレコーダにセットします. このとき, テープカウンタを0にしておけば, 後でどこからデータが入っているか一目でわかり, とても便利です.

(6) カセットテープのセットが終わったら, テープレコーダの録音ボタンと, PLAYボタンを押します. これらの準備が終わったら,

を入力します.

(7) を入力すると, すぐにカセットテープが回り始め, しばらくしてディスプレイに"Ok"と出力し, カセットテープは止まります.

これで, キーボードから入力した数値がカセットテープに書き込まれました.

2. データの読み込み

先ほどテープに書き込んだデータを, 今度は読み込んでみましょう.

(1) 次のプログラムを入力します.

プログラムの意味は次のとおりです.

(a) 画面に"push PLAY"と表示し, PLAYボタンを押すことを促します(100行).

```
100 PRINT "push PLAY"
110 OPEN "cas:suchi" FOR INPUT AS #1
120 INPUT #1,A,B
130 INPUT #1,C,D
140 PRINT A,B,C,D
150 CLOSE #1
160 END
```

(b) ファイルをオープンし，カセットテープからデータを入力するためのバッファを割り当てます(110行).

(c) INPUT#文を用いてデータを読み込みます(120，130行).

(d) データの読み込みが終わったら，CLOSE文でファイルを閉じます(150行).

(2) プログラムが入力できたら，データの書き込みを始めたところまでカセットテープを巻き戻します.

(3) カセットテープの準備ができたら実行させます.

run ⏎

(4) "push PLAY"というメッセージが現れたらPLAYボタンを押します．カセットテープが回り始めてしばらくすると，読み込まれたデータがディスプレイに表示されます.

3. データの型と数の一致

カセットテープにデータを書き込む，あるいは読み込む場合の注意事項として，書き込むときと読み込むときの〈データの型と数の一致〉があります．データとして文字を書き込んだのに，それを数値データとして読み込むことはできません．また，1つのPRINT文で4つのデータを書き込んでいるのに，INPUT文では2つしか読み込まないというわけにもいきません．ただし，データの型と数が一致していれば，読み込んだデータを代入する変数は，書き込んだときに使った変数と異なっていてもかまいません.

良い例1

書き込むとき

```
10 OPEN "cas:sample" FOR OUTPUT AS #1
          ⋮
100 PRINT #1,A,B,C
          ⋮
500 CLOSE #1
```

読み込むとき

データの型(数値)，数(3つずつ)が一致しているのでOKですね．

```
10 OPEN "cas:sample" FOR INPUT AS #1
          ⋮
150 INPUT #1,X,Y,Z
          ⋮
300 CLOSE #1
```

良い例2

書き込むとき

```
10 OPEN "cas:sample" FOR OUTPUT AS #1
          ⋮
100 PRINT #1,YAMA$,KAWA$
          ⋮
500 CLOSE #1
```

読み込むとき

これも，データの型(文字)，数(2つずつ)が一致しているのでOKです．

```
10 OPEN "cas:sample" FOR INPUT AS #1
          ⋮
200 INPUT #1,SORA$,UMI$
          ⋮
600 CLOSE #1
```

悪い例

書き込むとき

```
10 OPEN "cas:sample" FOR OUTPUT AS #1
          ⋮
100 PRINT #1,A,B,C,D
          ⋮
700 CLOSE #1
```

読み込むとき

```
10 OPEN "cas:sample" FOR INPUT AS #1
          ⋮
200 INPUT #1,A,B
210 INPUT #1,C,D
          ⋮
800 CLOSE #1
```

データの型(数値)は一致していますが，1つのPRINT文で4つのデータを書き込んだのに対し，読み込むときは一度に2個しか扱っていません．この場合，読み込むことはできません．

また，カセットテープ上にデータファイルをオープンして入出力を行う場合は，フロッピーディスクの場合と同じようにファイル名を付けることが可能ですが，フロッピーディスクのようにファイル名によってファイルを識別することはできません．

したがって，

OPEN "CAS:A" FOR OUTPUT AS #1

としてオープンし，データを書き込んだファイルを，

OPEN "CAS:B" FOR INPUT AS #1

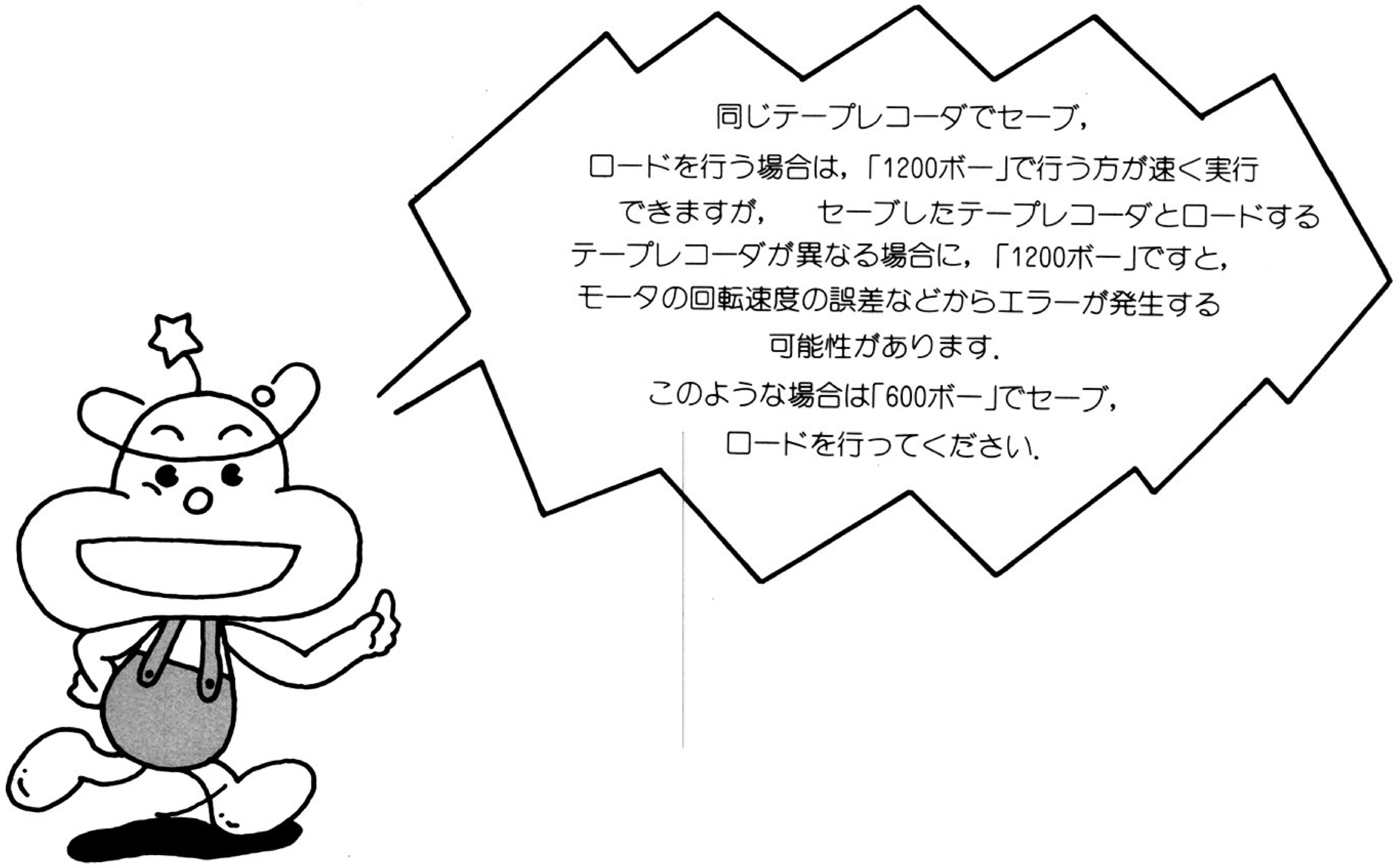
としてオープンし，データを読み出すこともできます．

4. カセットテープの転送速度

N₈₈-BASICでカセットテープを使用する場合に，「600ボー」と「1200ボー」の２つの転送速度を使用します．この指定は，デバイス名を指定すると，N₈₈-BASICが自動的に設定してくれます．

同じ長さのプログラムやデータをセーブしたりロードする場合，1200ボーですと600ボーの半分の時間で実行できます．しかし，600ボーでセーブしたプログラムやデータを1200ボーでロードすることはできません．また，逆に1200ボーでセーブしたプログラムやデータを600ボーでロードすることもできません．ですから，プログラムやデータをセーブする場合は，どちらの転送速度でセーブしたのかを記録しておいてください．

デバイス名	転送速度
cas: （cas1:）	1200ボー
cas2:	600ボー



第8章
割り込みキー

N88-BASIC/N88-日本語BASICは，
3種類の割り込みキーをサポートしています．
ここでは，これらの3種類の
割り込みキー(ファンクションキー，ヘルプキー，
ストップキー)をすべて使ったプログラムを例に，
それぞれの役割りと使い方を説明します．

このプログラムは，0から，入力された数値Nまでの2乗と平方根を求めるもので，各割り込みキーは次のように機能しています．

(1) ファンクションキー([f・1]・[f・2])

◦F・1——プリントオフ．このキーを押すと，以後画面への表示をしなくなります．

◦F・2——プリントオン．このキーを押すと，以後画面に表示します．

140行で処理ルーチンの開始行を定義し，170行でキーをオンしています．

(2) ヘルプキー([HELP])

メッセージの出力・入力の仕方，プログラム終了の仕方などのメッセージを表示します．

[HELP]でINPUT文実行中に割り込んだ場合，単にRETURN命令で復帰させるとINPUT文の次から再開することになり，正常に入力されないまま実行に移ってしまいます．そこで，処理ルーチンの中でそのフラグを返して，実行に入る前に判定しています(230行)．

(3) ストップキー([STOP])

ストップキーマスク．プログラム実行中に不注意で[↵]を押して実行が中断されないようにマスクしています．

マスクとはある処理機能，またはそれに対する要求を一時的に抑止することを意味します．

このように，それぞれの割り込みキーはその特徴を心得たうえで使ってください．

```
100 '--  KEY INTERRUPT sample program  --
110 '
120   WIDTH 80:CONSOLE 0,25,0,1
130   COLOR 7,0,0,7:CLS
140   ON KEY GOSUB *PRINTOFF,*PRINTON
150   ON STOP GOSUB *MASK
160   ON HELP GOSUB *MESSAGE
170   KEY(1) ON:KEY(2) ON
180   STOP ON:HELP ON
190 '
200 *ENTER
210   N=0:FLAG=0
220   INPUT "N=";N
230   IF FLAG<>0 THEN *ENTER
240   IF N<0     THEN *EXIT
250   IF N>100   THEN *ENTER
260   PRINT
270 '
280   HELP OFF:PSW=1
290   FOR I=0 TO N
300     PRINT "N=";I;
310     IF PSW=0 THEN PRINT:GOTO 340
320     PRINT TAB(15);"N*N =";I*I;
330     PRINT TAB(30);"SQR(N) =";SQR(I)
340   NEXT I:PRINT
350   HELP ON:GOTO *ENTER
360 '
370 *EXIT:'-- end routine --
380   KEY OFF:HELP OFF:STOP OFF
390   END
400 *PRINTOFF:'-- print off --
410   PSW=0:RETURN
420 *PRINTON:'-- print on --
430   PSW=1:RETURN
440 *MESSAGE:'-- message routine --
450   PRINT :PRINT "<< Instruction >>":PRINT
460   PRINT "N , N*N , SQR(N) ｦ ｹｲｻﾝ ｼﾏｽ｡"
470   PRINT "N ﾊ 0 ｶﾗ 100 ﾏﾃﾞ"
480   PRINT "N ﾆ 0 ﾐﾏﾝﾉ ｶｽﾞｦ ｲﾚﾙﾄ ｵﾜﾘﾏｽ｡"
490   FLAG=1:PRINT:RETURN
500 *MASK:'-- stop key mask --
510   RETURN
```

```
N=?
<< Instruction >>

N , N*N , SQR(N) ｦ ｹｲｻﾝ ｼﾏｽ｡
N ﾊ 0 ｶﾗ 100 ﾏﾃﾞ
N ﾆ 0 ﾐﾏﾝﾉ ｶｽﾞｦ ｲﾚﾙﾄ ｵﾜﾘﾏｽ｡

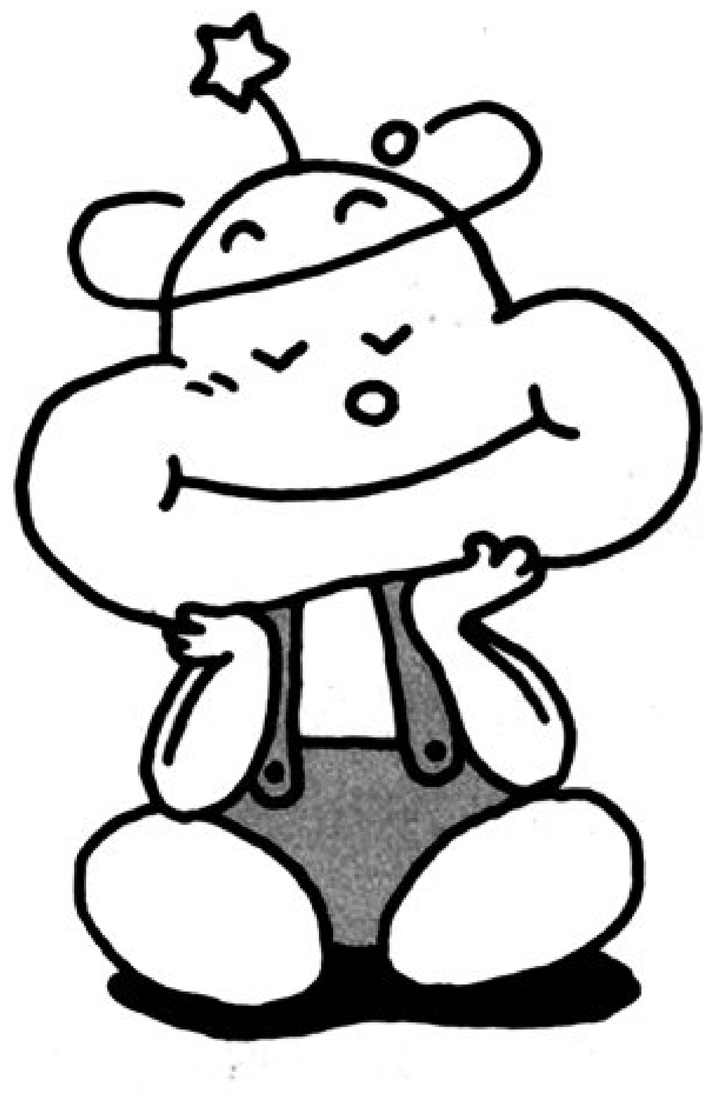
N=? 6

N= 0           N*N = 0        SQR(N) = 0
N= 1           N*N = 1        SQR(N) = 1
```

割り込みキーを機能的に使うとスマートなプログラムが書けますし，実行速度の向上にもつながります．

```
N= 2                N*N = 4             SQR(N) = 1.41421
N= 3                N*N = 9             SQR(N) = 1.73205
N= 4                N*N = 16            SQR(N) = 2
N= 5                N*N = 25            SQR(N) = 2.23607
N= 6                N*N = 36            SQR(N) = 2.44949

N=? -1
```



第9章 RS-232Cを使う

RS-232Cでは，通信機能として，ターミナルモードにより外部のコンピュータのターミナルとして使用するものと，BASICモードによりRS-232Cを一つのファイルとして扱い，BASICの管理下において使用するものの2つがあります．

ここでは，BASICモードによる使い方を説明します．

このプログラムは，PC-8801mkⅡMR 2台をケーブルで接続し，データを送受信するものです．

まず，送信側では初めに通信形式を設定します．各パラメータはTERMコマンドと同じです．次に文字変数に送信するデータを文字として与えます．120行で送信を行います．ファイル番号は，OPEN文で指定したファイル番号です．最後にファイルをCLOSEします．

一方，受信側では送信側と同じ通信形式を設定し，モードをINPUTにします．110行で送信されてきたデータを受信し，120行でそのデータを画面に表示します．そしてファイルをCLOSEします．

送信側

```
100 OPEN "COM:E73XS" FOR OUTPUT AS #1
110 A$="Personal Computer"
120 PRINT #1,A$
130 CLOSE
```

受信側

```
100 OPEN "COM:E73XS" FOR INPUT AS #1
110 INPUT #1,A$
120 PRINT A$
130 CLOSE
```

次に，RS-232Cによる割り込みを説明します．ON COM GOSUB文で割り込みが起こった場合の処理ルーチンを定義します．COM ONで割り込みを許可します．これ以後，RS-232Cから入力割り込みがあると，定義されている処理ルーチンへ制御を移します．RETURN文で中断したところから再開します．

このプログラムは送信，受信の両方に使用でき，割り込みによって受信されるよう

```
100 FALSE=0
110 TRUE=NOT FALSE
120 ECHO=TRUE
130 OPEN "COM:E73XS" AS #1
140 ON COM GOSUB *RECEVCHAR
150 COM ON
160 C$=INKEY$
170 IF C$<>"" THEN PRINT #1,C$;:IF ECHO THEN PRINT C$;
180 GOTO 160
190 *RECEVCHAR
200 IF LOC(1)=0 THEN RETURN
210 N$=INPUT$(LOC(1),#1)
220 PRINT N$;
230 RETURN
```

になっています．割り込みがない場合は160～180行の処理を繰り返し，キーボードからは1文字ずつ入力されるので，その文字をデータとして1文字ずつ送信し，また，入力がない場合も160～180行の処理を繰り返しますが，何も送信はしません．またこの間，割り込みが発生するとRECEVCHARのルーチンに飛んで，送信されてくる文字を画面に表示し，RETURN文で戻り，160～180行の処理を繰り返します．

第10章 デバッグ

プログラムが思ったとおりに動作しないことを「プログラムにバグ(虫)がいる」といい，それをなおすことをデバッグ(虫とり)といいます．ここでは，BASICでプログラムを作ったときのデバッグの方法について説明します．

1．TRONコマンドを使ってのデバッグ

デバッグの際，注目すべき点として，プログラムの実行順序と，変数の内容の変化の2つが考えられます．

まずプログラムの実行順序を調べてみましょう．これを行うにはTRONコマンドを実行します．TRONコマンドはダイレクトモード，プログラムモードの両方で実行できます．TRONコマンドを実行すると，プログラムが実行された順に行番号が表示されますので，それによってプログラムの実行順序を調べることができます．CTRL＋Sを押すと，プログラムの実行を一時停止，あるいは再開することができます．

そのようにして，実行順序がおかしい行を見つけたら，その1つ手前の行の内容をLISTコマンドによって表示させます．そして，その行で参照されている変数の内容の変化が指定したとおりになっているかを調べます．これを行うには，次のようにその変数に代入が行われた行(必ずしもLISTコマンドによって表示された行とは限りません)の前後にPRINT文を挿入します．

例

```
        ⋮
90 B=B+0.1
100 A=SIN(B)        ←—注目している行
110 CSC(A)=1/SIN(A)
        ⋮
```

上記のプログラムに次のようなPRINT文を挿入します．

```
95 PRINT "debug line #95;";A,B
105 PRINT "debug line #105;";A,B
```

このように変更したプログラムを実行して，変数の内容が正しく変化しているか調べます．

数値変数の場合，一度も代入が行われなかったときの内容は0になります．したがって，上記の挿入文によって変数の内容を表示したとき，0になっているものがあれば，表示以前の行で代入が正しく行われたか確認してください．

上記のような現象は，変数名にミススペルがあった場合によく起こります．

間違いを見つけ，変更することによって削除したい行がある場合は，DELETEコマンドを使わずREM文（'）を使用します．この理由は，もしその変更が間違いとわかったとき，元に戻すのを簡単にするためです．

例

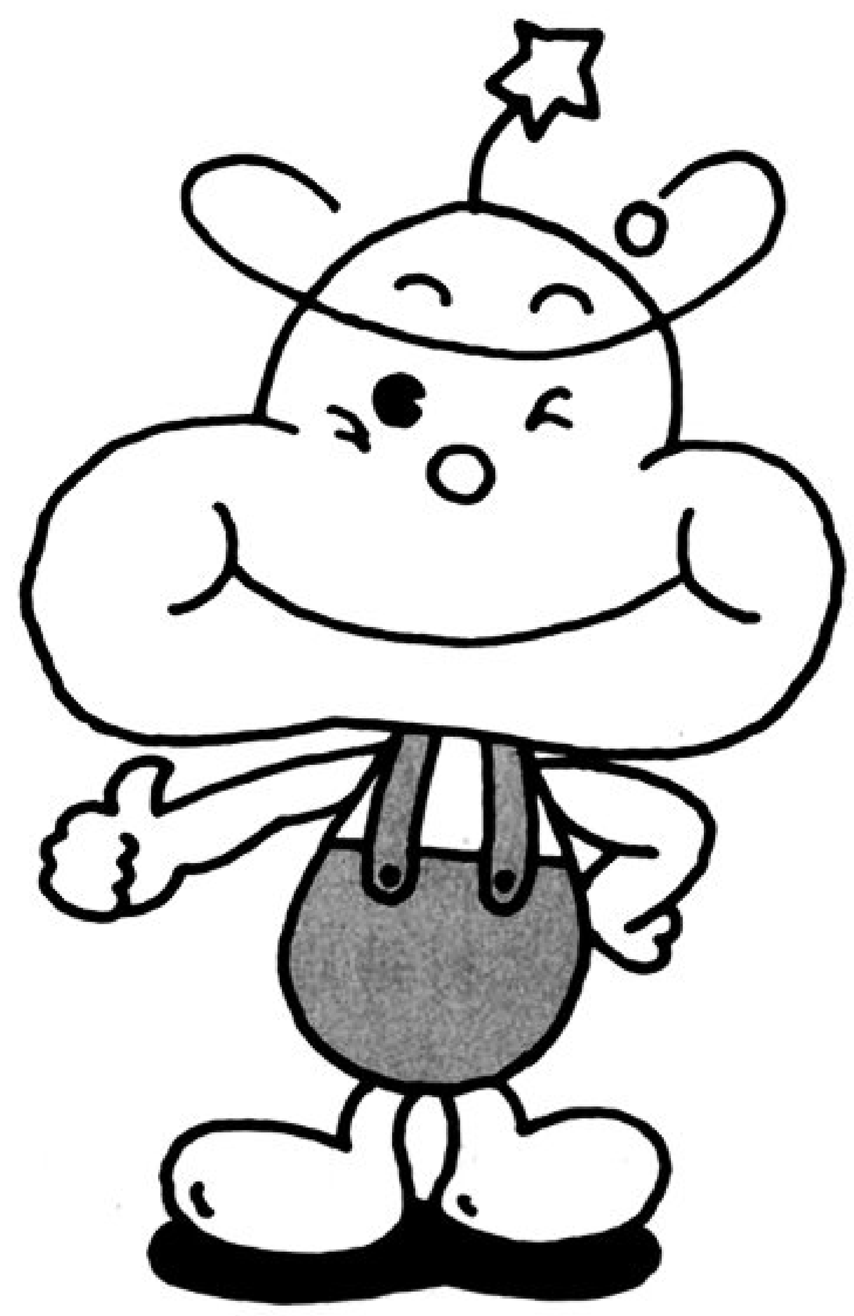
```
100 A=SIN(B)
        ↓
100 'A=SIN(B)
```

また，挿入したい行がある場合は，その行を挿入し，その後に新規追加した行であるむねをREM文（'）を使ってコメントとして残しておくと便利です．

例

```
100 A=SIN(B)
110 CSC(A)=1/SIN(A)
          ↓
100 A=SIN(B)
105 A=A+1 'bug fix
110 CSC(A)=1/SIN(A)
```

このようにしてデバッグを行い，完成したら，不要となった行やコメントを取り除きます．しかし，実行速度やプログラムサイズがそのプログラムに悪影響を与えない場合は，コメントは残しておいた方が後の変更やデバッグのために便利です．



2. スイッチを使ってのデバッグ

もっとデバッグしやすくするためには，プログラムそのものをデバッグしやすいよ

うに作ります．そのために，スイッチと呼ばれる変数を用意します．変数の値を0あるいは−1に設定し，その変数の値を切り替えることによってプログラムの実行順序を変えます．

スイッチはプログラムの最初の行で設定します．そして，プログラムの実行により一度代入されると，その値は以後変化しません．

たとえばDEBUGというスイッチを用意して，DEBUG＝−1のとき変数を表示するようにプログラムを作ると，デバッグを行うのに便利です．そして，デバッグが終わったらDEBUG＝0にしておきます．

またスイッチは，デバッグが終わっても，後でバグが見つかったときのために残しておいた方がよいでしょう．

例

```
10 FALSE=0
20 TRUE=NOT FALSE
30 DEBUG=TRUE 'debug if true
          ⋮
95 IF DEBUG THEN PRINT  "debug
   line #95; "; A, B
100 A=SIN(B)
105 IF DEBUG THEN PRINT "debug
    line #105; "; A, B
          ⋮
```

3. デバッグのために

デバッグを行う際の注意すべき点，参考点をあげました．

(1)　プログラムの見やすさを重視して，マルチステートメントは避ける．

```
10 INPUT "A="; A: INPUT "B="; B
20 C=A*B: PRINT C
         ↓
10 INPUT "A="; A
20 INPUT "B="; B
30 C=A*B
40 PRINT C
```

(2) 演算子の優先順位に気を付ける(自信がないときはとりあえず( )を付ける).

(3) 整数化する際のまるめ方に注意する(N_{88}-BASICでは，小数点以下は四捨五入されます).

(4) コロン(：)とセミコロン(；)，ピリオド(．)とコンマ(，)，ゼロ(0)とオー(O)，エル(l)とイチ(1)が間違っていないか注意を払う.

(5) 必要に応じてSTOP文を入れ，プログラムの実行順序や変数の内容を確認する(CONTコマンドを使用すると，STOP文の次の行から実行が再開できます).

(6) ファイルに出力を行うプログラムの場合，デバッグ中は画面に出力する.

```
10 OPEN "TEST.DAT" FOR OUTPUT
   AS #1
          ↓
10 OPEN "SCRN:" FOR OUTPUT
   AS #1
```

また，ファイルから入力する場合は，キーボードから入力を行ってデバッグする.

```
10 OPEN "TEST.DAT" FOR INPUT
   AS #1
          ↓
10 OPEN "KYBD:" FOR INPUT AS
   #1
```

(7) ON ERROR GOTOを使ったプログラムをデバッグする場合, ON ERROR GOTOが実行される状態だとエラー処理を行う必要が生じるため，デバッグがやりにくくなる. デバッグ中は，ON ERROR GOTO文をコメント文(REM文を使用)に変更しておく.



第11章 機械語プログラムを呼ぶ

N88-BASIC/N88-日本語BASICには，機械語で書かれたプログラムを呼び出す機能としてUSR関数とCALL文が用意されています．ここでは機械語プログラムを作成する際の注意と，USR関数，CALL文との引数の受け渡し方について説明します．なお，ここでの説明を正しく理解するためには，機械語プログラムに関する知識を必要とします．

1．メモリの配置

機械語ルーチンをモニタにより作成したり，ファイルからロードする際には，そのためのメモリ領域を確保する必要があります．この機械語プログラム用のメモリ領域は，BASICの使用する領域と重複してはなりません．さもないと，BASICの動作が異常になったり，逆にBASICの動作により準備した機械語プログラムが破壊されたりします．メモリ領域の確保は，CLEAR文によりBASICの使用するメモリの上限を設定することにより行います．指定された上限以上の領域はBASICは利用しなくなりますので，機械語プログラム領域として用いることができます(CLEAR文参照)．

2．機械語プログラムの準備

確保した領域に機械語プログラムを準備する主な方法としては，次の3つがあります．

(1)　モニタを用いる

MONコマンドにより機械語モニタに移り，モニタの機能を利用してプログラムを書き込む(付録1　機械語モニタ参照)．

(2)　BLOADにより機械語プログラムを読み込む

すでにBSAVEによりセーブされている機械語プログラムをロードする．

(3) POKEによりプログラムを書き込む

比較的短いプログラムであれば，DATA文により機械語プログラムをデータの型で表現しておき，それをREAD文で読み出した後，POKE文により書き込む．

また，機械語プログラムを作成する際には次の点に注意してください．

(a) BASICに制御を戻すためには，プログラムはリターン命令(RET)を用いなければなりません．

(b) 機械語ルーチンに引き渡された引数が文字列であった場合には，引数の値は次のような3バイトからなるストリングディスクリプタと呼ばれる領域のアドレスとなります．

(i) ストリングディスクリプタの最初のバイトは，その文字列の長さ(0から255)を保持します．

(ii) 2番目のバイトは，文字列の置かれている先頭アドレスの下位8ビットを保持します．

(iii) 最後のバイトは，文字列の置かれている先頭アドレスの上位8ビットを保持します．

機械語ルーチンでは，この3バイトのストリングディスクリプタを書き換えてはなりません．また，ストリングディスクリプタによって指されるストリングの内容は変更することはできますが，その長さを変更してはなりません．

(c) 機械語ルーチンからBASICに戻るときには，呼び出されたときとスタックポインタ(SP)の値が等しくなっていなければなりません．

(d) 機械語ルーチンが呼び出されたとき，スタック領域としては8レベル(16バイト)分だけ利用できるように設定されています．もし機械語ルーチン内でより大きなスタックを必要とする場合には，ルーチン内で独自にスタックを用意しなければなりません．またこのとき，BASICに戻るときにスタックポインタをもとに戻すことを忘れてはなりません．



3. USR関数の呼び出し

USR関数は

USR〔〈番号〉〕(〈引数〉)

という書式によって呼び出されます．ただし〈番号〉は0から9までの数値で，引数は任意の数値式または文字式です．USR関数は書式上〈引数〉を必要としますので，呼び出されたルーチンが引数を必要としない場合でも，ダミーの引数を与える必要があります．

USR関数が呼び出され，対応する機械語ルーチンに制御が移ったとき，Aレジスタは，渡された引数の型を示す値を持っています．その値と意味は次のようになっています．

Aの値	引数の型
2	整数型
3	文字列型
4	単精度型
8	倍精度型

引数が文字列の場合には，〔DE〕レジスタペアが3バイトから成るストリングディスクリプタを指しています．

引数が数値の場合には，〔HL〕レジスタに浮動小数点アキュムレータ(FAC)と呼ばれる8バイトの領域の5バイト目のアドレスが入っています．実際の引数は，このFACの内に各型に応じて以下のように格納されて渡されます．

● 整数型のとき

FACの5バイト目(〔HL〕レジスタペアが指しているところ)に引数の下位8ビット，6バイト目に上位8ビットが入ります．

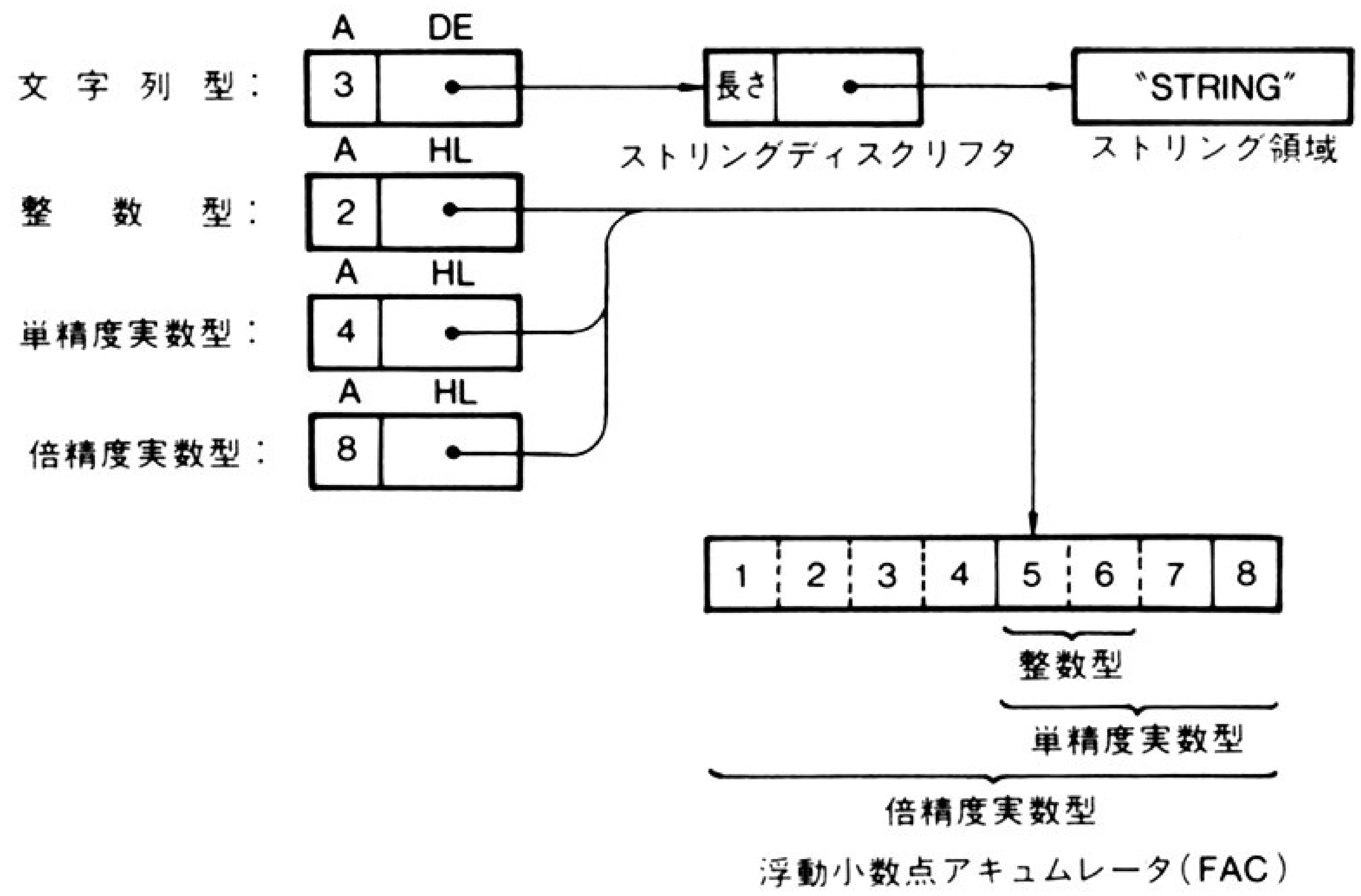
● 単精度実数型のとき

FACの8バイト目は指数部になっており，(指数−128)の値が入ります．小数点は仮数部の最上位ビットの左にあると想定します．7バイト目は仮数部の最高部7ビットを保持します．このバイトの最上位ビットは符号を示し，0で正，1で負を表します．6バイト目と5バイト目は，それぞれ仮数部の中部と最低部の8ビットを保持します．

●倍精度実数型のとき

5バイト目から8バイト目までは，単精度実数型と同じように指数部と仮数部の上位3バイトが入ります．1バイト目から4バイト目には，仮数部の下位4バイトが入ります．

USR関数がBASICに値を返すには，FACで求めた関数値を設定してやることで行えます．通常，返す値は，引数として渡した値と同じ型となります．



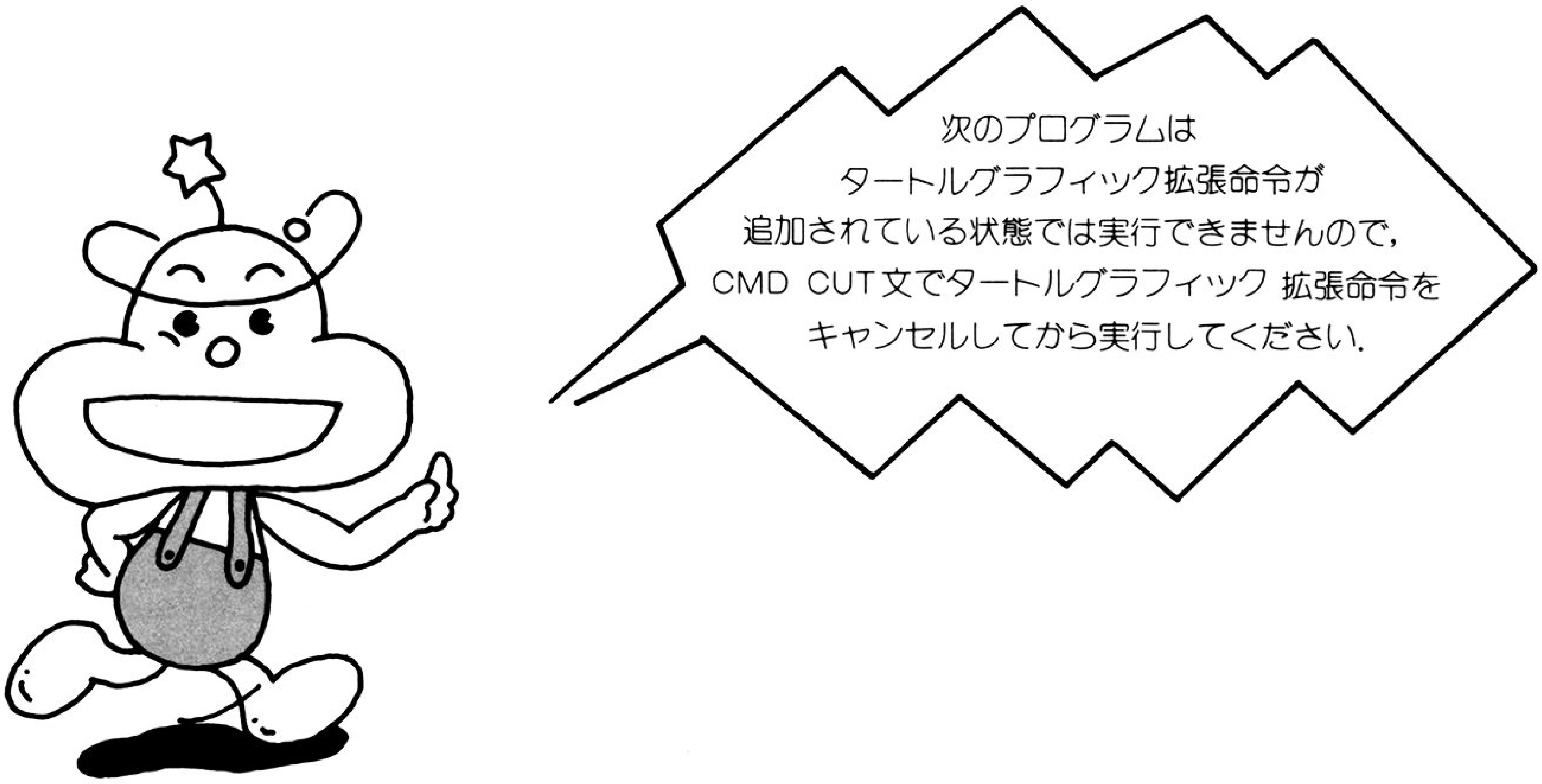
サンプルプログラム

USR関数の例として，文字列中に含まれる小文字の数を返す関数を示します．ここではDATA文により機械語プログラムを用意しておき，POKEにより書き込む方法により機械語プログラムを用意しています．

機械語ルーチンでは，まずAレジスタの値を調べて，与えられた引数が文字型かどうかを調べています．ここでは，文字型でない場合はそのままBASICに戻ることにしています．

引数が文字型であった場合には，〔DE〕レジスタにより渡されたストリングディスクリプタの番地をもとに，文字列の長さと，実際に文字列の置かれている番地を求めます．

その後，ストリングを最初から1文字ずつ調べていき，小文字であった場合には，ANI命令によりD_5ビットを0にすることにより大文字に変換し，ストリングを変更します．このようにユーザ関数は与えられた引数自身を変更するために，USR0の呼び出し前後ではA$の内容が変化することになります．




```
100 '  USR function sample program
110 '
120 CLEAR ,&HE000  'reserve program area
130 MACHINE=&HE000
140 DEF USR0=MACHINE
150 '
160 FOR ADR=MACHINE TO MACHINE+30
170   READ BYTE:POKE ADR,BYTE  'poke ML program
180 NEXT ADR
190 '
200 INPUT A$
210   B$=USR0(A$)
220   PRINT B$
230 GOTO 200
240 END
250 '
260 '  machine language program data
270 '
280 DATA &HFE,&H03       :'CPI  3      ;ARGUMET MUST STRING
290 DATA &HC0            :'RNZ         ;IF NOT,JUST IGNORE
300 DATA &HEB            :'XHCG
310 DATA &H56            :'MOV  D,M    ;D HAS LENGTH
320 DATA &H23            :'INX  H
330 DATA &H7E            :'MOV  A,M
340 DATA &H23            :'INX  H
350 DATA &H66            :'MOV  H,M
360 DATA &H6F            :'MOV  L,A    ;HL=POINTER TO STRING
370 DATA &H14            :'INR  D      ;NULL STRING  ?
380 '              LOOP:
390 DATA &H15            :'DCR  D      ;DECREMENT LENGTH
400 DATA &HC8            :'RZ          ;FIND STRING END ?
410 DATA &H7E            :'MOV  A,M    ;LOAD A CHAR.
420 DATA &HFE,&H61       :'CPI  61H    ;CHAR>'a' ?
430 DATA &HDA,&H1B,&HE0:'JC   NOTLOW;No.
440 DATA &HFE,&H7B       :'CPI  7BH    ;CHAR<='z'  ?
450 DATA &HD2,&H1B,&HE0:'JNC  NOTLOW;No.
460 DATA &HE6,&HDF       :'ANI  0DFH   ;CONVERT TO UPPER
470 DATA &H77            :'MOV  M,A    ;SET IT
480 '            NOTLOW:
490 DATA &H23            :'INX  H
500 DATA &HC2,&H0B,&HE0:'JMP  LOOP
```

4. CALL文

機械語ルーチンはCALL文を用いても呼び出すことができます.

引数を持たないCALL文は機械語のCALL命令と同等であり，指定された番地からのルーチンが呼び出されます．BASICに戻るには，USR関数と同じようにRET命令で戻ります.

CALL文が引数を持っている場合には，レジスタ中に，引数で指定された変数の置かれている番地(VARPTR関数で求められる番地と同一のもの)が準備された上で，機械語ルーチンが呼び出されます．引数の番地情報を渡す方法は，引数の数によって異なります.

(1) 引数が3個以下のとき

1番目の引数の番地は〔HL〕レジスタに，

2番目の引数の番地は〔DE〕レジスタに(もしあるならば)，

3番目の引数の番地は〔BC〕レジスタに(もしあるならば)

準備されます.

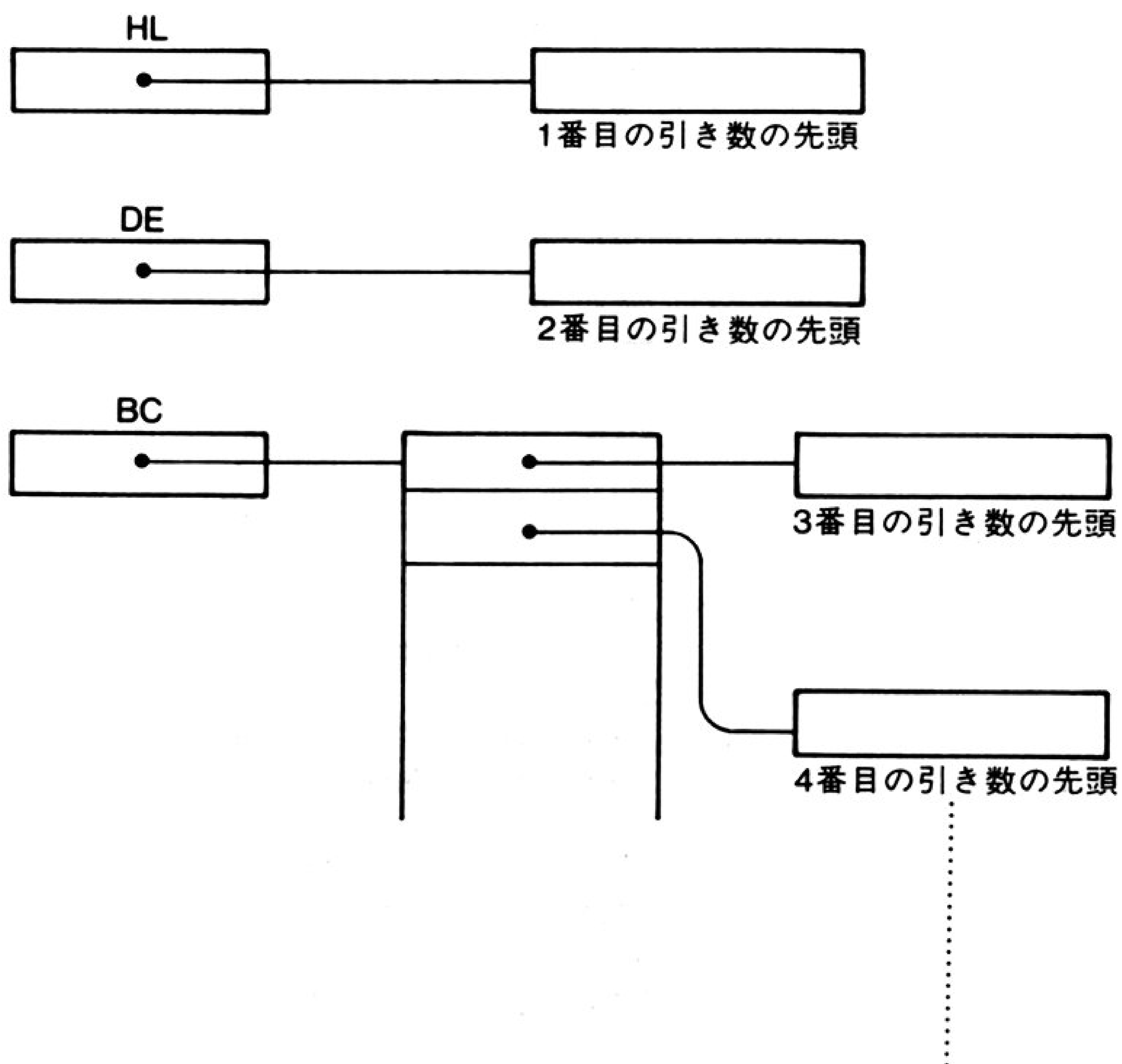
(2) 引数の数が4個以上のとき

1番目の引数の番地は〔HL〕レジスタ，

2番目の引数の番地は〔DE〕レジスタに

準備されます.

3番目以降の引数については，それらの番地が順番に並べられた表が作成され，〔BC〕レジスタにその表の番地が準備されます.



CALL文を用いる際には，次の事項に注意してください.

● CALL文が渡す引数の番地は，USR関数の場合のFACの番地ではなく，VARPTR関数の値と同じ，変数値の置かれている番地です.

● CALL文では，複数の引数を与えることができます．また，与えられた引数の番地を介してBASICに値を返すことができます.

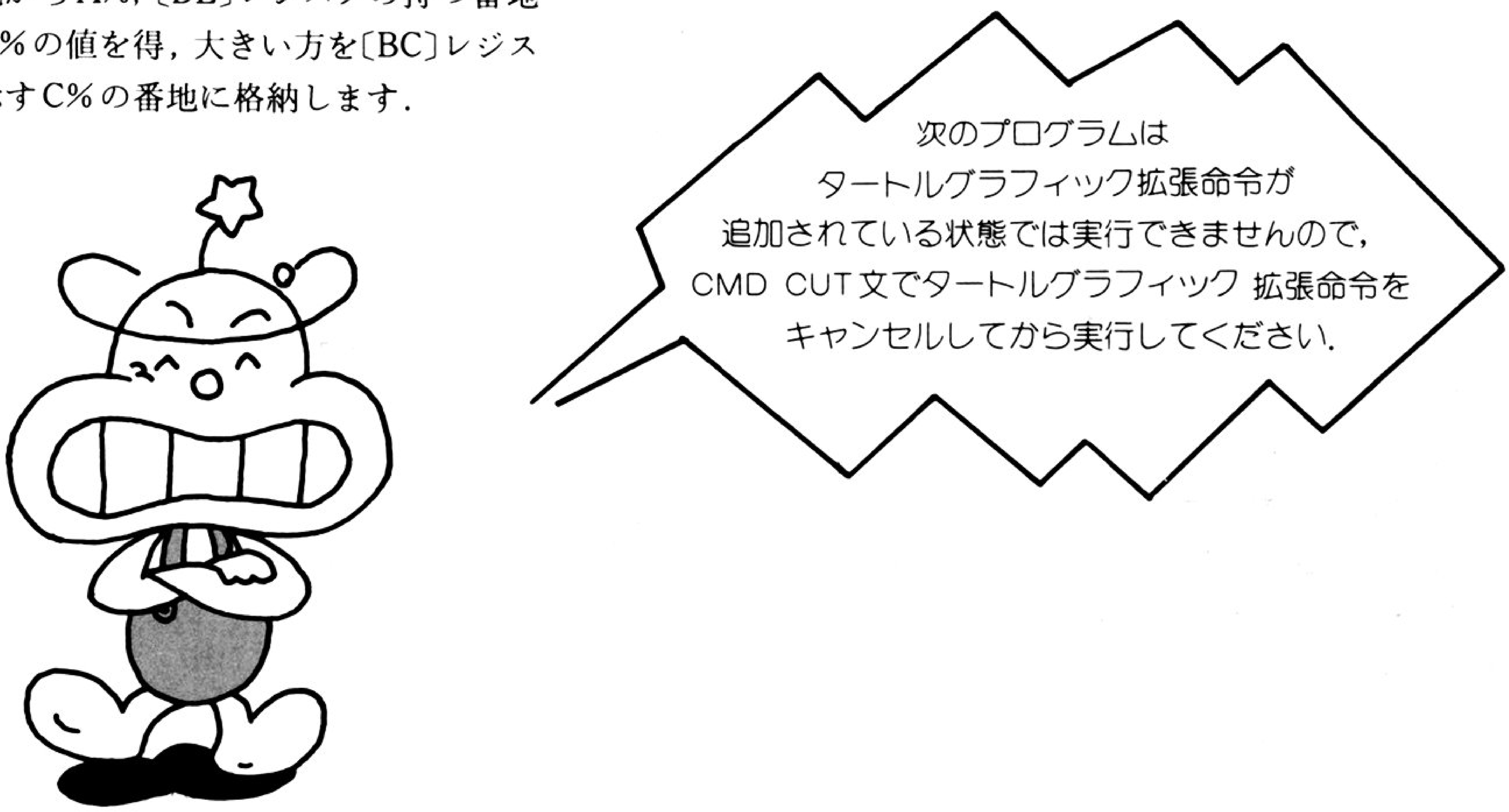
●引数が文字型の場合には，準備される番地はストリングディスクリプタの番地です．

● CALL 文は，引数の個数や型については何ら情報を与えません．したがって，これらを一致させることに注意しなければなりません．

サンプルプログラム

CALL 文の例として，2つの数の最大値を与えるルーチンを示します．A%, B%により渡された2つの数のうちの大きい方の値をC%に代入します．

機械語ルーチンでは〔HL〕レジスタの持つ番地からA%,〔DE〕レジスタの持つ番地からB%の値を得，大きい方を〔BC〕レジスタの示すC%の番地に格納します．




```
100 '  CALL statement sample program
110 '
120 CLEAR,&HE000  'reserve program area
130 MAX=&HE000
140 '
150 FOR ADR=MAX TO MAX+45
160   READ BYTE:POKE ADR,BYTE  'poke ML language
170 NEXT ADR
180 '
190 C%=0
200 INPUT A%,B%
210   CALL MAX(A%,B%,C%)
220   PRINT C%
230 GOTO 200
240 END
250 '
260 '  machine language program data
270 '
280 '                      MAX:
290 DATA &HDB,&H71        :'IN   71H
300 DATA &HF5             :'PUSH PSW
310 DATA &H3E,&HFF        :'MVI  A,0FFH
```

```
320 DATA &HD3,&H71       :'OUT   71H         ;SELECT N88 MAIN ROM
330 DATA &HCD,&H29,&HE0:'CALL GETVAL      ;GET A% VALUE
340 DATA &HEB           :'XCHG
350 DATA &HCD,&H29,&HE0:'CALL GETVAL      ;GET B% VALUE
360 DATA &HE5           :'PUSH H
370 DATA &HB7           :'ORA  A           ;CLEAR CARRY
380 DATA &H7B           :'MOV  A,E         ;A%-B%
390 DATA &H95           :'SUB  L
400 DATA &H7A           :'MOV  A,D
410 DATA &H9C           :'SBB  H
420 DATA &H2E,&HFF      :'MVI  L,0FFH
430 DATA &HE2,&H1A,&HE0:'JPO  *+4         ;CHECK V FLAG
440 DATA &H2C           :'INR  L
450 DATA &HAD           :'XRA  L
460 DATA &HE1           :'POP  H           ;ASSUME B%<A%
470 DATA &HF2,&H20,&HE0:'JP   *+4
480 DATA &HEB           :'XCHG             ;A%<=B%
490 DATA &H7D           :'MOV  A,L         ;STORE INTO C%
500 DATA &H02           :'STAX B
510 DATA &H03           :'INX  B
520 DATA &H7C           :'MOV  A,H
530 DATA &H02           :'STAX B
540 DATA &HF1           :'POP  PSW
550 DATA &HD3,&H71      :'OUT  71H
560 DATA &HC9           :'RET
570 '             GETVAL:
580 DATA &H7E           :'MOV  A,M
590 DATA &H23           :'INX  H
600 DATA &H66           :'MOV  H,M
610 DATA &H6F           :'MOV  L,A
620 DATA &HC9           :'RET
```

CALL文使用上の注意

CALL文を使ってROM内ルーチンを使用する場合，以下の手順を必ず踏んでください．

〈BASIC〉

```
CALL ××××
```

〈機械語〉

```
××××: IN 71H      ┐
      PUSH PSW    │ メインROM
      MVI A, FFH  │ をセレクトする
      OUT 71H     ┘
        ⋮
   (ユーザプログラム)
        ⋮
      POP PSW     ┐ ROMの状態を
      OUT 71H     │ 元へ戻す
      RET         ┘
```

例

"A"という文字を画面へ出力する

〈BASIC〉

```
100 CLEAR, &HCEFF
200 ADR=&HCF00
300 CALL ADR
400 END
```

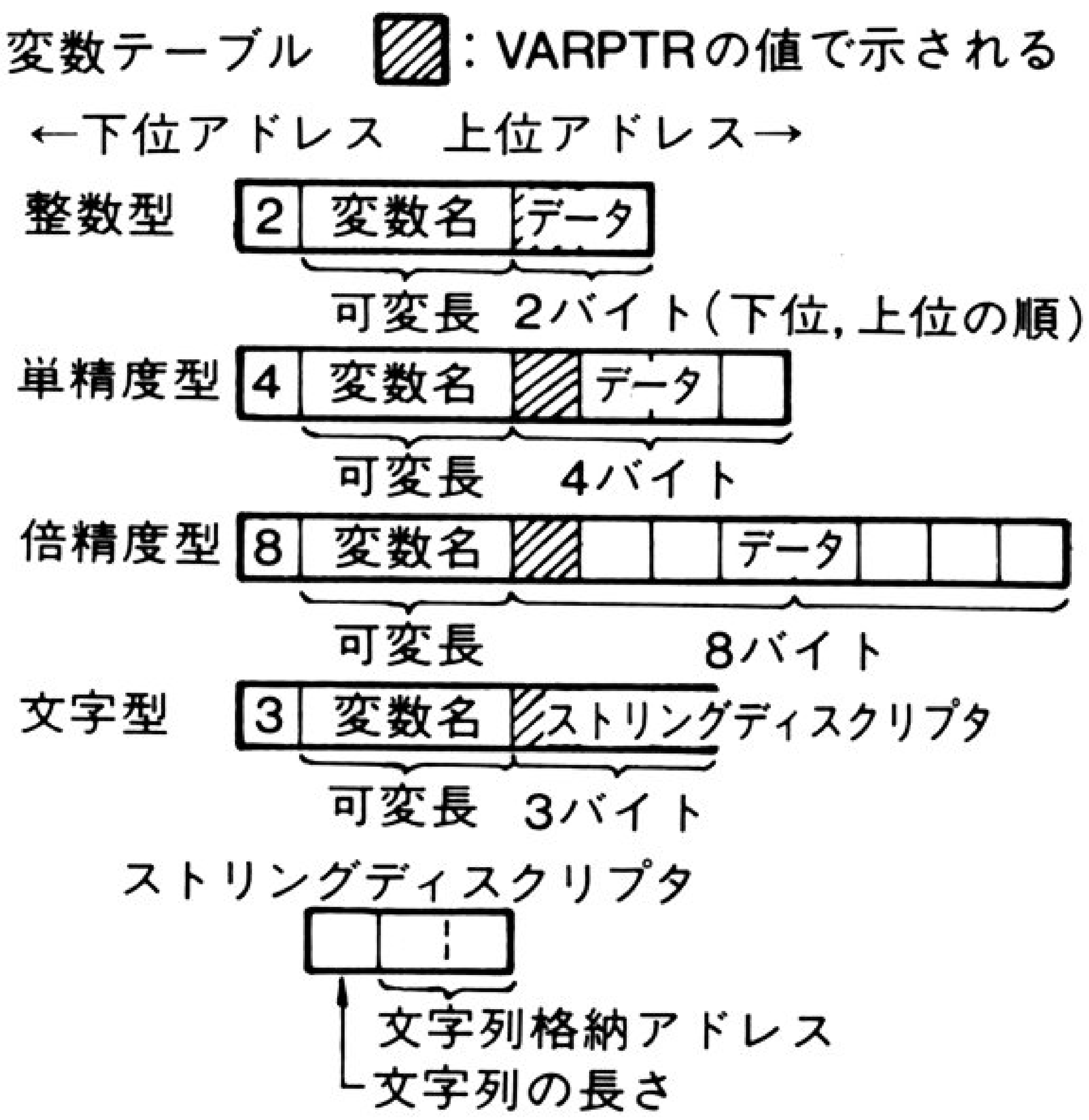
〈機械語〉

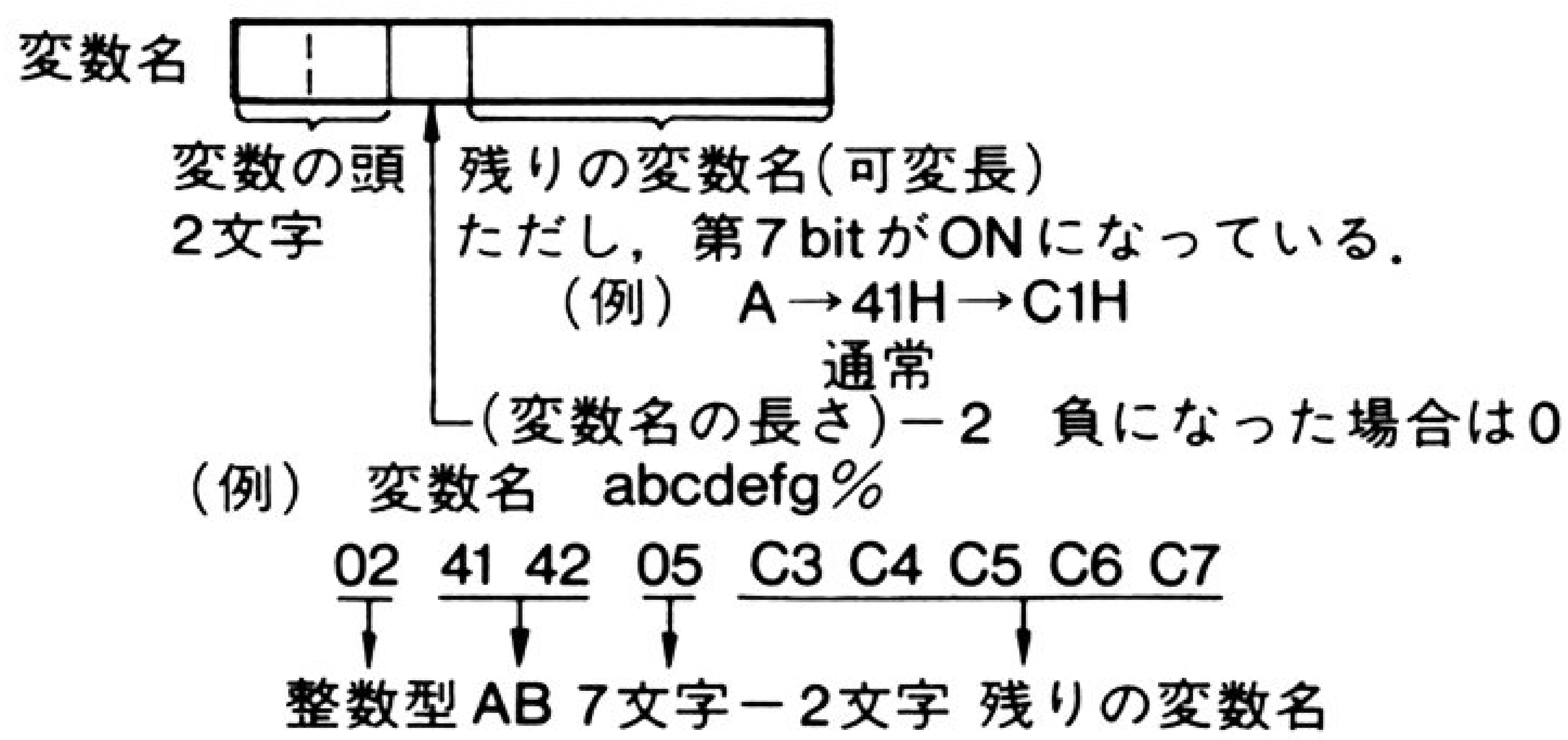
```
CF00: IN 71H
      PUSH PSW
      MVI A, FF
      OUT 71H
      MVI A, 41H  ┐ "A"を表示
      RST 3       ┘
      POP PSW
      OUT 71H
      RET
```

5. VARPTR

VARPTR関数は，指定された変数，バッファの置かれている番地を返します．

VARPTR関数の引数として変数名が与えられたときには，その変数の値の最下位バイトが置かれている番地を返します．上位バイトは，下図に示すようにこの番地以降に入っています．また，文字変数名が指定された場合には，ストリングディスクリプタの先頭番地が返されます．





変数として配列要素が指定された場合には，その要素の置かれている番地が返されます．

引数としてバッファ番号が与えられた場合には，そのバッファ領域の番地を返します．この領域の最初の9バイトはBASIC作業用の領域となっており，ファイルとの入出力に用いられる256バイトの領域は，VARPTR関数の値に9を加えた番地から始まります．

付　録

付録では，機械語モニタ，
ターミナルモードの使い方，およびシンセサイザICの
構造について説明します．
いずれも，自分でプログラムを
作ったりする人にとって必要となることですから，
はじめてパーソナルコンピュータに触る人，
市販のソフトウェアを利用するだけの人は，
読む必要はありません．

機械語でプログラムしょうとする人のために

付録1．機械語モニタ

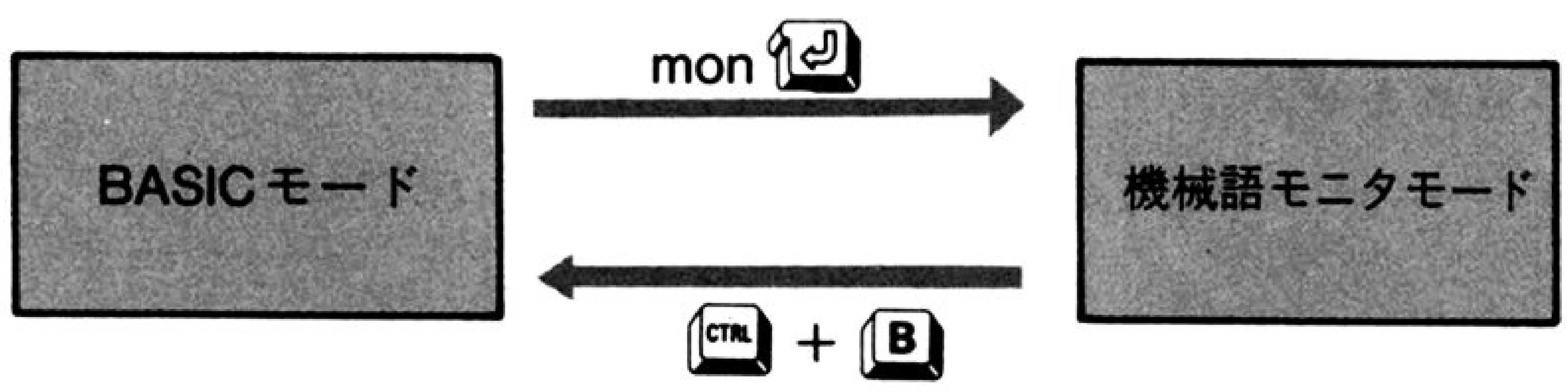
この章は，PC-8801MKⅡMRで直接機械語を扱う場合に使用する機械語モニタについて説明します．

PC-8801MKⅡMRには，独立したI/Oコントロールルーチンを持つ強力な機械語モニタが準備されています．この機械語モニタの主な特長は次のとおりです．

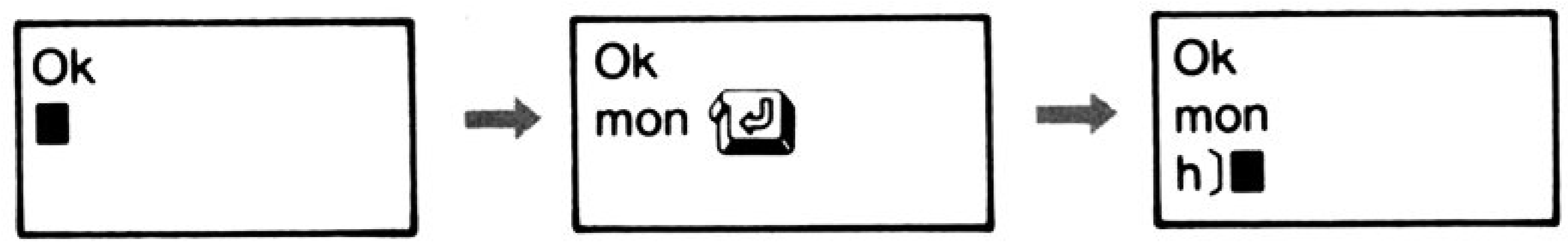
1. ミニアセンブラ機能を持っている．
2. 数値を16進数，8進数のどちらでも入力，表示が可能．
3. I/Oポートへの入出力が可能．
4. 逆アセンブル機能を持っている．
5. CPUのレジスタの値の表示，変更が可能．
6. カセットテープにも，フロッピーディスクにもロード，セーブが可能(カセットテープを使う場合は，CMTインタフェースボードが必要)．
7. スクリーンエディタによるメモリの内容の変更ができる．
8. コマンドの入力形式を知りたいときのために，HELPコマンドが用意されている．

付録1.1　機械語モニタの動かし方

機械語モニタは，BASICモードから機械語モニタモードに切り替えて使用します．



(1)　BASICモードから機械語モニタモードへの切り替え方法

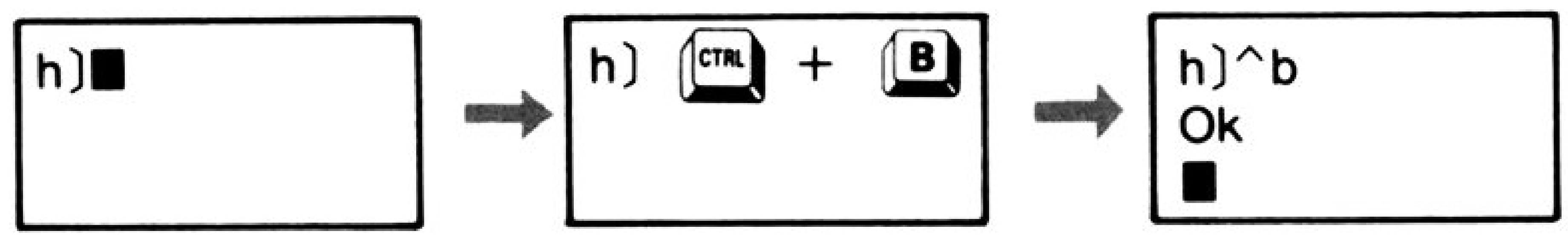


BASICコマンドが入力可能.

MONコマンドを入力.

機械語モニタモードになり，h〕が表示される.
機械語モニタのコマンド入力が可能.

(2)　機械語モニタモードからBASICモードへ戻るための方法



機械語モニタのコマンドが入力可能.

CTRL と B とを同時に押す.

BASICモードになり，Okが表示されて，次の行にカーソルが現れる.
BASICコマンドが入力可能.

付録1.2 機械語プログラムのメモリ配置

BASICを使っているときのメモリマップは，右図のようになっています.

XXXX, ZZZZの番地は，選ばれているBASICのモードなどによって変わります．機械語プログラムをメモリ上に書き込む前に，次のようにして番地を調べてください.

● XXXXの調べ方

```
print □ hex$(peek(&HE7E9)*256+
peek(&HE7E8))↵
```

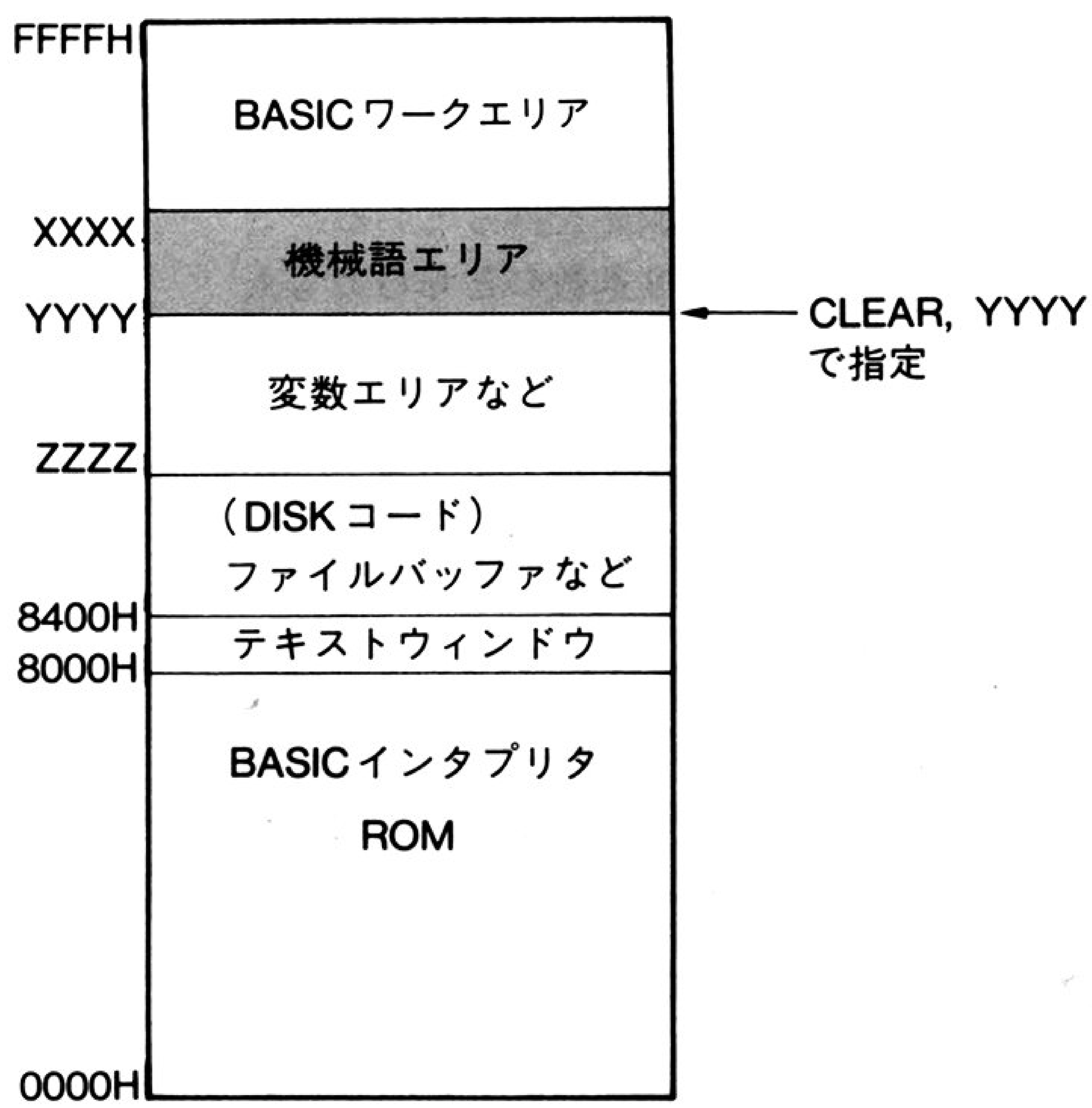
● ZZZZの調べ方

```
print □ hex$(peek(&HEB1C)*256+
peek(&HEB1B))↵
```

機械語のプログラムを作る場合は，CLEAR命令で，機械語エリアを確保しておかなければなりません.

```
clear □ ,YYYY↵
```

とすることで，図の色のついた部分を機械語エリアとして使うことができるようになります．それ以外の部分は，BASICが使っていますのでその内容を変更するとBASICモードに戻ったときの動作が保障されません.



付録1.3 コマンド使用上の規則

(1) 機械語モニタのコマンドの英文字は，大文字でも小文字でも入力できます．

(2) 機械語モニタではBASICとは異なって，スクリーンエディタの機能がありません（ただし，Eコマンドを参照）．また，[INS DEL]を使って入力した文字を消して修正することもできません．誤ったコマンドを入力したことに気づいたときは，[STOP]または[CTRL]＋[C]で誤ったコマンドを取り消したのち，改めて正しいコマンドを入力してください．

(3) 機械語モニタでは，入力できるキーは少数に限られていて，誤ったキーを入力すると？が表示されます．

(4) 機械語モニタで扱う数値形式には，16進数モードと8進数モードがあります．この切り替えは，Bコマンドで行います．システムが起動した直後は，16進数モードになっています．

たとえば，16進数を入力するときに，BASICのように&Hを付けて入力する必要はありません．

① 16進数モード
　アドレス　　：4桁　0～FFFF
　メモリの内容：2桁　0～FF

② 8進数モード
　アドレス　　：6桁　0～177777
　メモリの内容：3桁　0～377

アドレスの場合は，入力された数値の終わりから16ビットが有効になります．メモリの内容の場合は，終わりから8ビットが有効になります．

また，機械語モニタの各コマンドで，アドレスやメモリの内容を入力する場合，加減の演算機能が使用できます（ただし，カッコや累乗は使えません）．

例

```
h] s9000 + 100 [RETURN]
9100 00 -
```

(5) 機械語モニタによって，カセットテープを使用するには，Rコマンド，Vコマンド，Wコマンドを用います．このとき，ファイル名の指定の方法により，転送速度が決まります．

(6) 機械語モニタモードでのディスプレイ画面

① BASICモードから機械語モニタモードへ切り替えた直後は，その直前のWIDTH文によって設定された状態が引きつがれます．

② テキスト画面のスクリーンウィンドウによっては，Eコマンドは動作しない場合があります．

このときは，機械語モニタモードに入る前に，

console␣0,〈17以上の値〉↵

を実行しておいてください．

付録1.4 コマンドの説明

(1) A(アセンブル)

Intel 8080ニーモニックで入力した1行のテキストをアセンブルし，できた機械語をメモリに格納します．そのために，Aコマンドは，

a〔〈格納開始番地〉〕

のように，普通は機械語の格納を開始する番地を指定します(格納開始番地の指定を省略した場合は，その前に実行したAコマンドの実行終了番地の次の番地となります)．

例

h〕a9000 ↵

Aコマンドを入力すると，格納開始番地を表示して，ニーモニックの入力待ちになります．そこで，Intel 8080ニーモニック命令を入力し，↵ を入力すると，その1行をアセンブルし，できた機械語をアドレスのすぐ横に表示し，そのアドレスに格納します．そして，その次の行に，次に機械語を格納する番地を表示し，ニーモニック命令の入力待ちになります．

一般形式　　注．〔〈ファイル名〉〕は省略可

R〔1:/2:〕〔〈ファイル名〉〕

W〔1:/2:〕〔〈ファイル名〉〕,〈セーブ開始アドレス〉,〈セーブ終了アドレス〉

V〔1:/2:〕〔〈ファイル名〉〕

ファイル名	転送速度
〔〈ファイル名〉〕 1:〔〈ファイル名〉〕	1200 ボー
2:〔〈ファイル名〉〕	600 ボー



```
h〕a9000 ↵
9000 3E 01 mvi a,1 ↵
```

入力したニーモニック命令にエラーがあって，？が表示された場合には，もう一度アドレスを指定しなくても，コマンドだけを入力すれば，エラーを起こしたときのアドレスから実行を開始します．

```
h〕a9000 ↵
9000 3E 01 mvi a,1 ↵
9002       mvl b,1 ↵
?
h〕a ↵
9002
```

このアセンブラは，すべて ↵ が入力されるまで，読み込みを続けます．また [INS DEL] が使えますから，間違えて入力した文字を訂正できます．このアセンブラから抜け出すには，アドレスが表示されたあと ↵ を入力するか，[STOP] または [CTRL]＋[C] を入力します．

このアセンブラで使用できるニーモニックは，Intel 8080ニーモニック命令とZ80ニーモニック命令の中のレラティブジャンプ命令です．Z80のレラティブジャンプ命令の場合は，絶対アドレスを入力すれば，アセンブラが相対アドレスに変換してくれます．

Z80のレラティブジャンプ命令の入力形式は，右のとおりです．

Z80	モニタで入力するとき
jr 〈アドレス〉	jmpr 〈アドレス〉
jr z,〈アドレス〉	jrz 〈アドレス〉
jr nz,〈アドレス〉	jrnz 〈アドレス〉
jr c,〈アドレス〉	jrc 〈アドレス〉
jr nc,〈アドレス〉	jrnc 〈アドレス〉
djnz 〈アドレス〉	djnz 〈アドレス〉

(2) B(ベース)

数値を取り扱う形式を設定するコマンドです．数値をキーボードから入力したり，ディスプレイに表示する場合に，16進数として扱うか8進数として扱うかを決めます．BASICモードからモニタへ切り替えたときは，16進数モードになっていて，コマンド入力待ちになると，

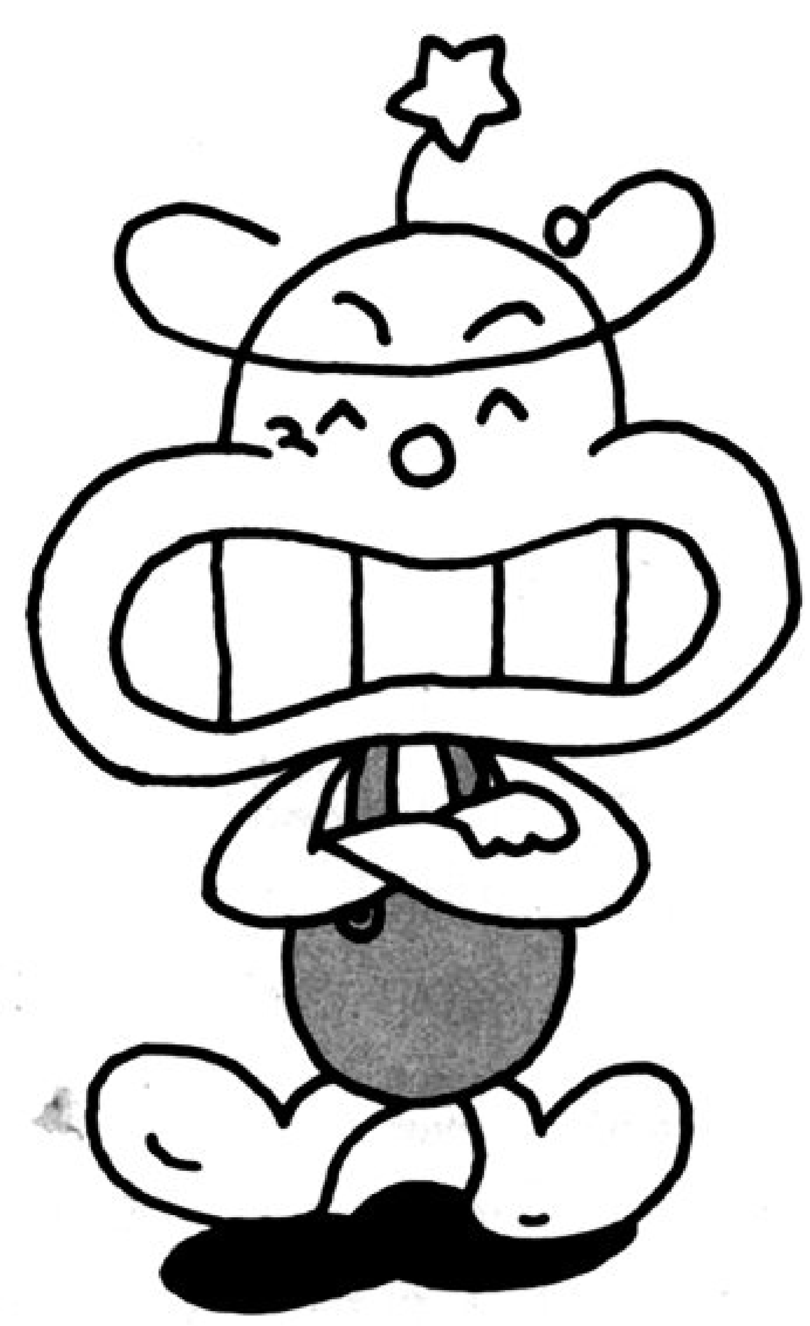
```
h〕
```

が表示されます．8進モードに切り替えるには，bqと入力します．

8進数モードになるとq〕が表示されます．

```
h〕bq ↵
q〕
```

再び16進数モードへ切り替えるには，



bhを入力します．

q〕bh [RETURN]

h〕

(3) D(ダンプメモリ)

メモリの内容をディスプレイに表示するコマンドです．16進モードになっている場合は，その数に対応するアスキー文字も表示されます．さらに，プリンタスイッチでプリントモードが指定されていれば，結果がプリンタへも出力されます(プリンタスイッチおよびプリントモードについては，(11) P(プリンタスイッチ)を参照してください)．Dコマンドは次の形で入力されます．

d〔<表示開始アドレス>〕〔,<表示終了アドレス>〕

ですから，実際の入力は，次の形になります．

(i) **d9000,9030** [RETURN]
(ii) **d9000** [RETURN]
(iii) **d,9030** [RETURN]
(iv) **d** [RETURN]

(i)の場合が標準で，9000H番地から9030H番地までのメモリの内容が表示されます．(ii)のように<表示開始アドレス>だけ指定した場合は，<表示開始アドレス>から，16バイト分のメモリの内容が表示されます．逆に(iii)のように<表示終了アドレス>だけ指定した場合は，その前に実行されたDコマンドの最終のアドレスの次のアドレスから，<表示終了アドレス>までのメモリの内容を表示します．(iv)のように<表示開始アドレス>も<表示終了アドレス>も指定しない場合は，このコマンドが入力される前に実行されたDコマンドの最終のアドレスの次のアドレスから16バイト分のメモリの内容が表示されます．

Dコマンドを実行中に，その実行を中止したい場合は，[STOP]または[CTRL]+[C]を入力します．画面にメモリの内容を表示している場合，途中で表示を一時停止させるには，[CTRL]+[S]を入力します．再び表示を再開させる場合には，[STOP]と[CTRL]+[C]以外のキーのどれかを入力します．

また，Dコマンド終了後に ⏎ を入力すると(iv)と同じ動作を行います．

(4) E（エディトメモリ）

Eコマンドは，スクリーンエディタの機能を用いて，スクリーンに表示されたメモリの内容を変更するコマンドです．Eコマンドは次の形式で入力します．

e〔〈開始アドレス〉〕

〈開始アドレス〉を省略した場合は，最後にEコマンドで内容を変更したメモリのアドレスが〈開始アドレス〉となります．

Eコマンドを実行して，メモリの内容をディスプレイに表示した後は，カーソル移動キー ↑ ↓ ← → と ROLL UP ROLL DOWN を使って変更したいメモリの内容のところへカーソルを移動させ，新しい数値を入力します．画面上で変更された数値のとおりに，メモリの内容も変更されています．エディトメモリを終了するときは，STOP または ESC を入力します．

(5) F（フィルメモリ）

アドレスで指定した範囲のメモリの内容を，指定した定数の値に置き換える命令です．コマンドの形式は次のようになっています．

f〈開始アドレス〉,〈終了アドレス〉,〈定数〉

Fコマンドでは，各パラメータを省略することはできません．コマンドが正常に実行されたかどうかは，Dコマンドを用いて確認します．

```
h〕f9000,9010,0 ⏎
h〕d9000,9010 ⏎
9000 00 00 00 00 00 00 00 00
9008 00 00 00 00 00 00 00 00
9010 00
h〕
```

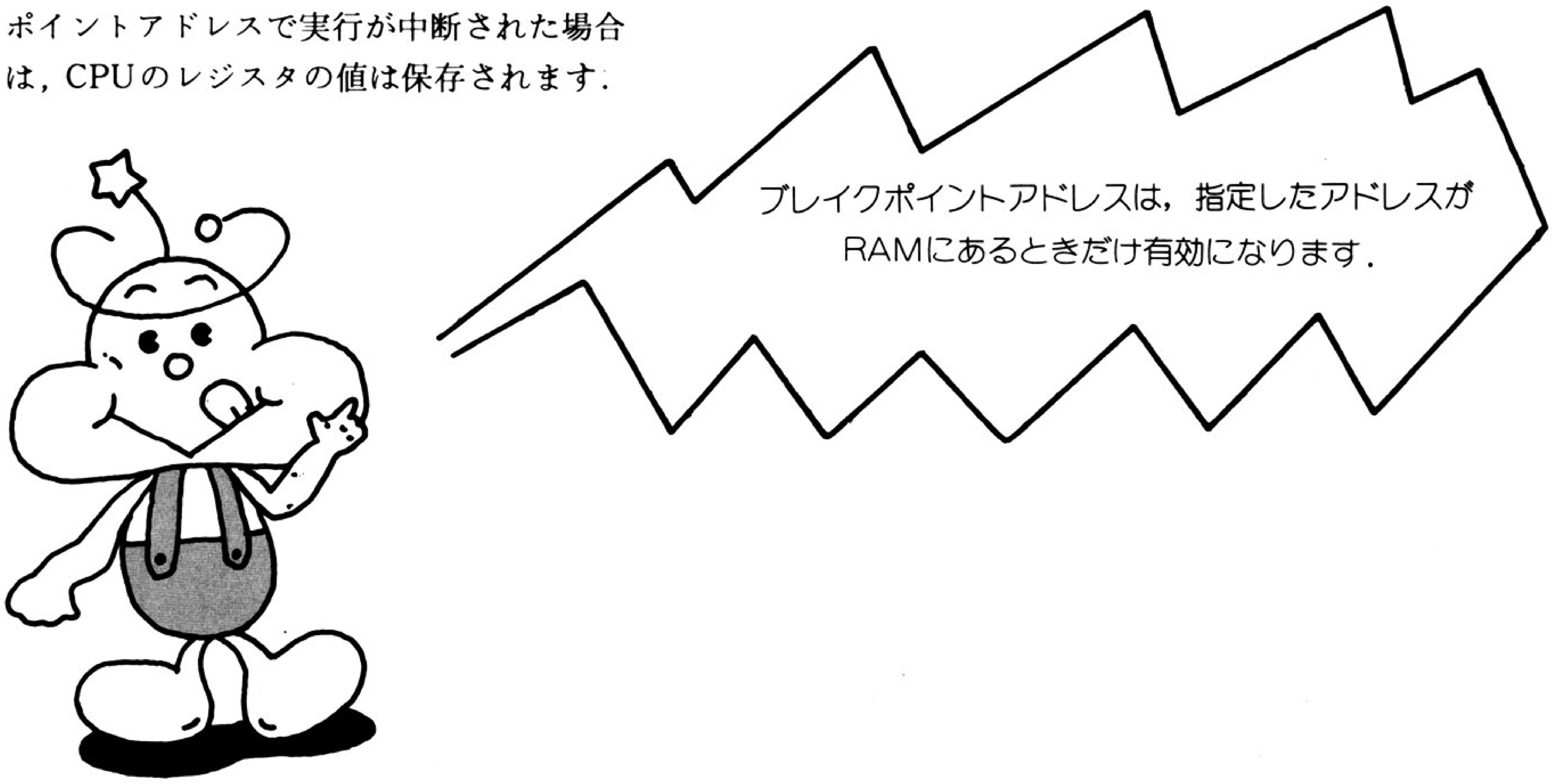
(6) G（ゴー）

指定したアドレスにジャンプして，そこからプログラムの実行を開始します．このとき，2つまでブレイクポイントアドレスを指定できます．ブレイクポイントアドレスとは，プログラムの実行中，実行アドレスがブレイクポイントアドレスとなったと

き，実行を中止して，コマンドの入力待ち状態にさせるアドレスです．

g〔〈実行開始アドレス〉〕〔，〈ブレイクポイントアドレス#1〉〕〔，〈ブレイクポイントアドレス#2〉〕

〈実行開始アドレス〉を省略した場合，最後に実行したブレイクポイントアドレスが〈実行開始アドレス〉になります．ブレイクポイントアドレスで実行が中断された場合は，CPUのレジスタの値は保存されます．



(7)　I（インプット）

入力ポートの値を読み，画面に表示します．ポートアドレスを入力した後，スペースを入力すれば，その行に入力ポートの値を表示し，⏎ を入力した場合は，次の行に入力ポートの値が表示されます．

i〈ポート　アドレス〉⏎

（または，スペースキー）

（Oコマンドの項を参照にしてください．）

(8)　L（ディスアセンブル）

メモリイメージをIntel 8080ニーモニックに逆アセンブルして表示するコマンドです．さらに，Z80のレラティブジャンプ命令も使用できますが，アドレスは，絶対アドレスで表示されます．Lコマンドの形式は次のようになっています．

l〔〈逆アセンブル開始アドレス〉〕〔，〈逆アセンブル終了アドレス〉〕

〈逆アセンブル開始アドレス〉が省略された場合は，最後に逆アセンブルされたアド

レスの次のアドレスが〈逆アセンブル開始アドレス〉になります．また，〈逆アセンブル終了アドレス〉が省略された場合は，〈逆アセンブル開始アドレス〉＋15が〈逆アセンブル終了アドレス〉となります．

Lコマンドの実行中に実行を中断したい場合は，[STOP]または[CTRL]＋[C]を入力します．実行を一時中断したい場合は，[CTRL]＋[S]を入力します．実行を再開するには[STOP]，[CTRL]＋[C]以外の任意のキーを入力します．また，Lコマンドで表示されるニーモニックのオペランドは，現在，指定されている数値モードで表示されます．

さらにLコマンド終了後に[RETURN]を入力すると，〈逆アセンブル開始アドレス〉と〈逆アセンブル終了アドレス〉を省略したときと同じように，〈逆アセンブル終了アドレス〉から16バイト分のメモリイメージを逆アセンブルします．

(9) M（ムーブメモリ）

Mコマンドは，メモリ内である位置から別の位置へ，メモリ領域のブロック転送を行います．Mコマンドは，次の形式で入力します．

m〈転送するメモリ領域の先頭番地〉，
〈転送するメモリ領域の最終番地〉，
〈転送先の先頭番地〉

たとえば，9000H番地から9100H番地のメモリ内容をA000H番地からA100H番地へ転送する場合は，

h］m9000，9100，a000 [RETURN]

と入力します．この後，Dコマンドを実行すれば，転送できたことを確認できます．

(10) O（アウトプット）

このコマンドは，指定したポートアドレスにデータを出力します．Oコマンドの入力形式は，

o〈ポートアドレス〉，〈データ〉

となります．

IコマンドおよびOコマンドは，自作のボードを機械語プログラムでコントロールする場合など，特殊な用途に使用するコマンドです．

(11)　P（プリンタスイッチ）

Dコマンド，Lコマンド，CTRL-Dコマンドを実行した結果を，プリンタに出力するかどうかを決定します．

モニタに入ったときは，プリンタスイッチはOFFになっていて，Dコマンド，Lコマンド，CTRL-Dコマンドを実行しても，結果はディスプレイに表示されるだけです．このときPコマンドを入力すると，プリンタスイッチはONになり，以後，実行されるDコマンド，Lコマンド，CTRL-Dコマンドの結果は，プリンタにも出力されます．

プリンタスイッチがONになるとプロンプト（入力促進記号）はh），あるいはq）になります．

プリンタスイッチをONから再びOFFにするには，再度Pコマンドを入力します．

(12)　R（リードテープ）

カセットテープからデータをロードするコマンドです.* Rコマンドは，次の形式で入力します．

r〔〈ファイル名〉〕

ファイル名を指定した場合は，カセットテープのデータのファイル名をチェックし，指定したファイル名が見つかると，そのデータをメモリにロードします．このとき，指定したファイル名以外のファイル名が見つかった場合は，

Skip　XXXX

とディスプレイに表示し，指定したファイル名が見つかるまで読み続けます．指定したファイル名が見つかった場合は，

Found　XXXX

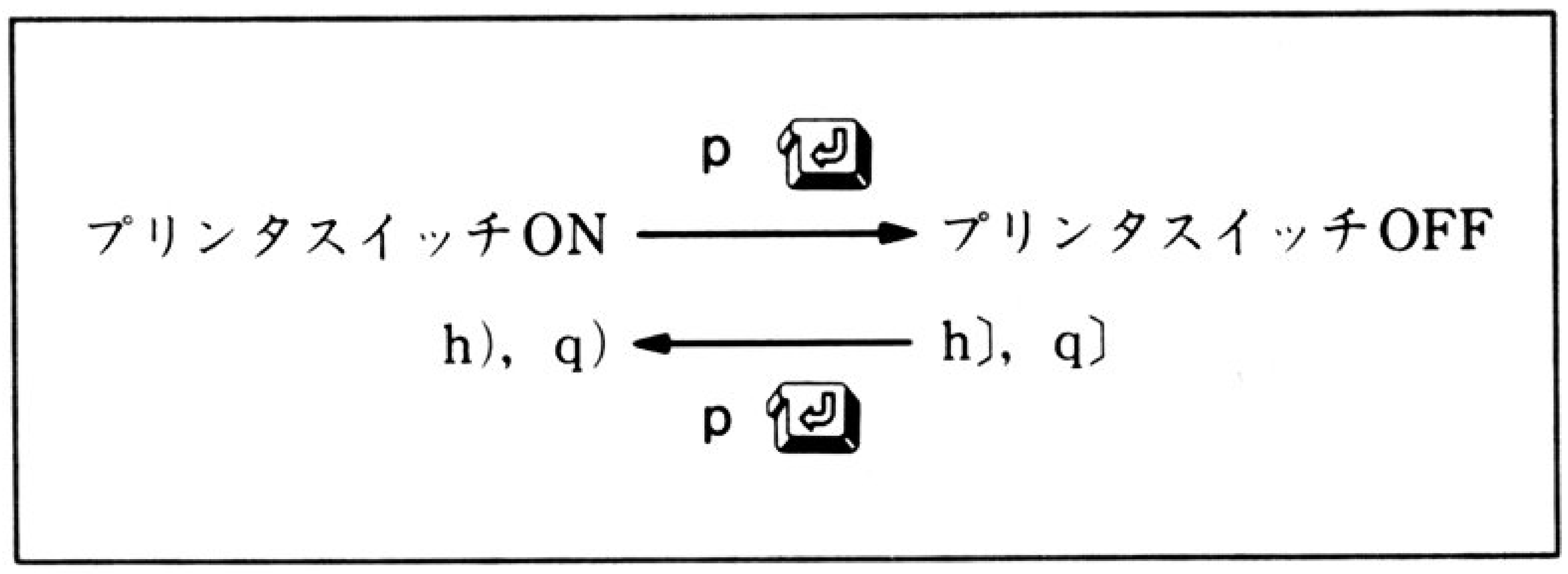
とディスプレイに表示し，その下にロードしているデータを表示します．

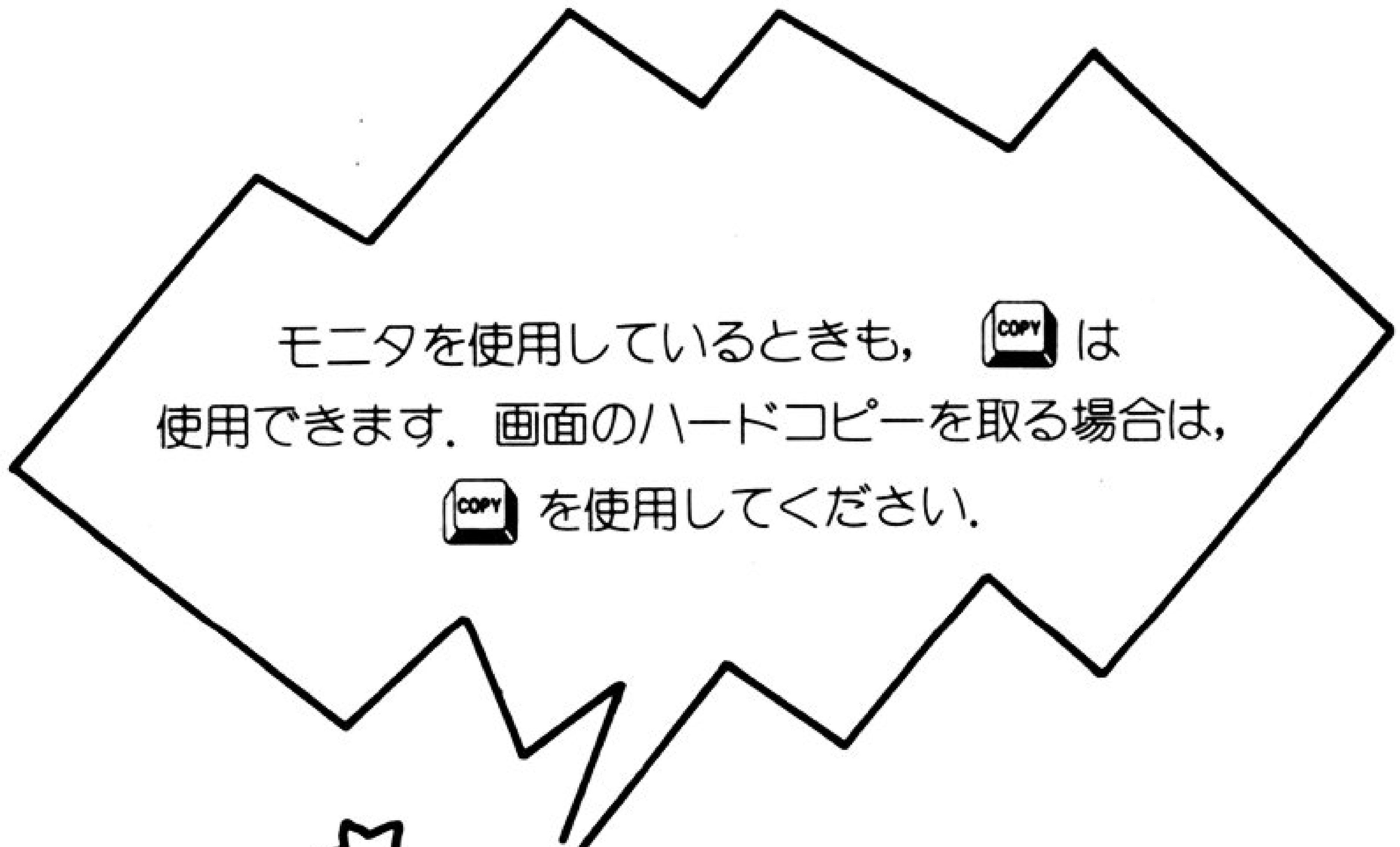
ファイル名を指定しない場合は，最初に見つかったファイルをロードします．

ファイル名は，6文字以下です．6文字を超えた場合は，最初の6文字が有効になります．

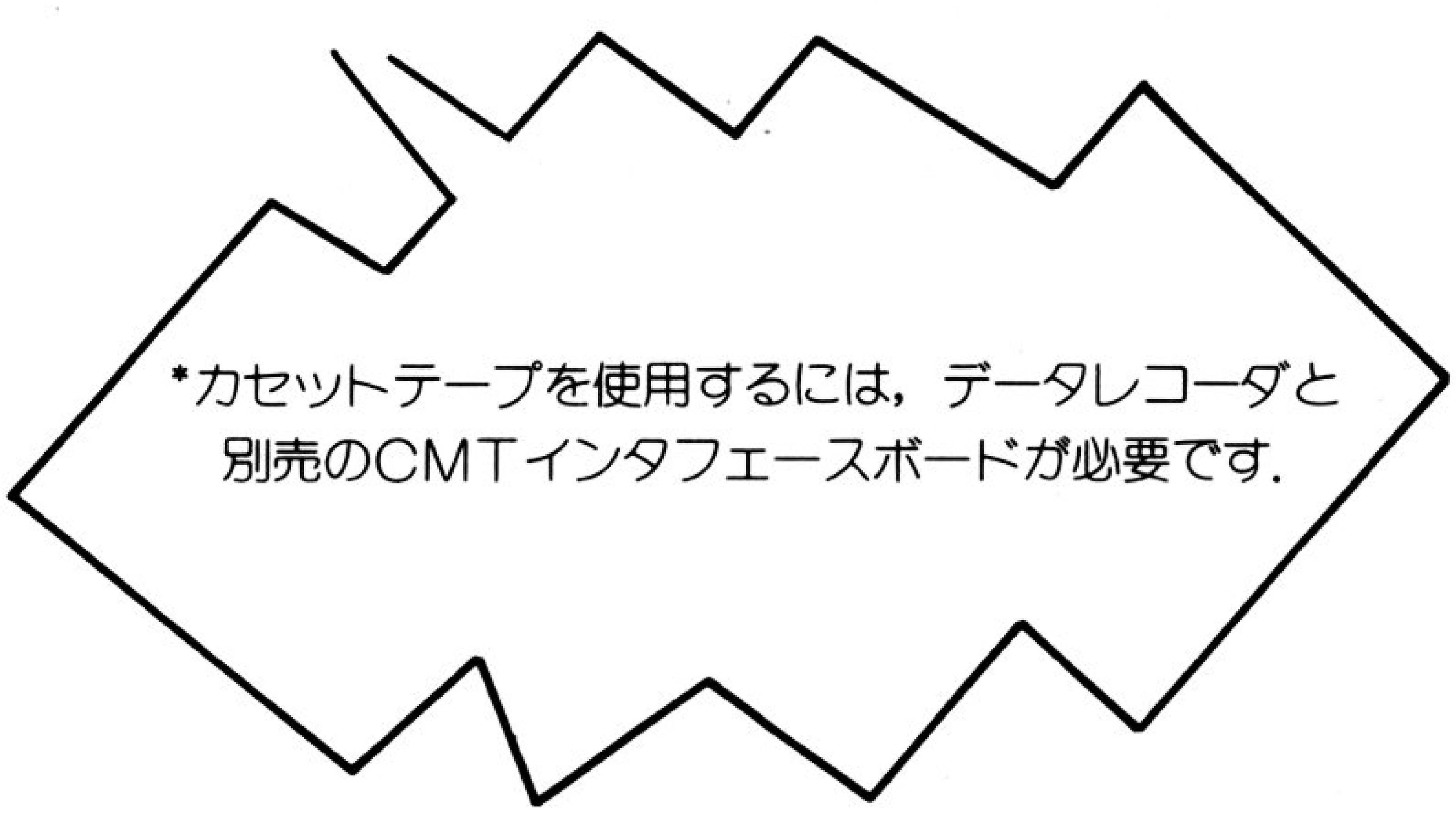
(13)　S（セットメモリ）

Sコマンドは，メモリの内容を確かめた









り，変更するコマンドです．Sコマンドは次の形式で入力します．

s〔〈開始アドレス〉〕⏎

（または，スペースキー）

Sコマンドで，〈開始アドレス〉を指定した後 ⏎ を入力すると，次の行にアドレスとその内容を表示し，数値の入力待ちになります．スペースキーを入力すると，改行せずにメモリの内容を表示し，数値入力待ちになります．

入力する数値は，16進モードならば，2桁の16進数を入力しますが，2桁以上の数値を入力した場合は，最後の2桁が有効となります．数値を入力した後スペースキーを入力すれば，メモリに値をセットし，次のアドレスのメモリの内容を表示します．また，数値を入力した後，⏎ を入力すれば，新しい値がセットされ，コマンド入力待ちになります．もし，メモリの内容を変更しないならば，数値を入力せずにスペースキーを入力すれば，次のアドレスに移ります．アドレスを1つ前に戻したい場合は，CTRL＋Bまたは←を入力します．

⒁　TM（テストメモリ）

N_{88}-BASICで使用されるユーザーズメモリ64Kバイトと，グラフィック用VRAM 48Kバイトをテストするコマンドです（N_{88}-日本語BASICで使われる拡張RAM 128Kバイトのテストは行いません）．テストは次の順序で行います．

① モニタに入る前に

screen 0，2 ⏎

を入力する．

② モニタへ入る．

mon ⏎

③ tm ⏎

④ 次の順でチェックします．

(i) テキストVRAM

(ii) 変数・ワークエリア

(iii) グラフィック用VRAM

(iv) テキストエリア

このテストには，V2モードでは約10分間，V1標準モードでは約30分間かかります．テストが無事に終了すると，



test complete !

というメッセージが現れますから，リセットボタンを押してください．

エラーが起きた場合，(i)で起きたときは，ブザー(beep音)が鳴り続けます．(ii)以後の場合は，エラーが起きた番地を表示します．

⒂ V(ベリファイテープ)

このコマンドは，カセットテープにセーブされたデータと，メモリの内容とが一致しているかどうかを比較するコマンドです.* Vコマンドは，次のように入力します．

v〔〈ファイル名〉〕

動作はRコマンドとほぼ同じで，〈ファイル名〉を指定した場合は，指定したファイルとメモリの内容を比較し，〈ファイル名〉を指定しなかった場合は，最初に見つかったファイルと，メモリの内容を比較します．比較した結果，カセットテープの内容とメモリの内容が一致した場合は，コマンド入力待ち状態となり，一致しなかった場合は，？を表示してコマンド入力待ちになります．

⒃ W(ライトテープ)

メモリの内容をカセットテープへセーブするコマンドです.* Wコマンドは次の形式で入力します．

w〔〈ファイル名〉〕，〈セーブ開始アドレス〉,〈セーブ終了アドレス〉

8000H番地から，8500H番地の内容を"abc"というファイル名でカセットテープにセーブする場合

h〕wabc , 8000 , 8500

⒄ X(イグザミンレジスタ)

Xコマンドは，CPUの全レジスタの内容の表示，および変更を行うコマンドです．

x〔〈レジスタ名〉〕

〈レジスタ名〉は，

a , b , d , h , a' , b' , d' , h' , s , p , x , y , i

です．a'のようにダッシュ記号がついているレジスタ名は，裏レジスタを指します．

〈レジスタ名〉を指定しない場合は，全レ



ジスタの内容を表示して実行を終了します．〈レジスタ名〉を指定した場合は，そのレジスタの内容を表示し，新しい値の入力待ちになります．新しい値を入力して，[↵] を入力すると，コマンド入力待ち状態になり，スペースキーを入力すると，次のレジスタの処理へと移ります．新しい値を入力しない場合も，[↵] を入力すれば，コマンド入力待ち状態となり，スペースキーを入力すると，次のレジスタの処理へと進みます．

各レジスタは，Iレジスタを除いて，すべてのレジスタは16bitのペアレジスタとして扱われます．ですから，8bit分のレジスタだけ変更する場合は，変更しない残りの8bit分のデータも入力してください．また，フラグレジスタで，フラグを立てないときは－を入れます．

⒅ [HELP] または [CTRL] ＋ [A]（ヘルプ）

[HELP] または [CTRL] ＋ [A] を入力すると，モニタで使用できるコマンドの種類とそのコマンドのパラメータがディスプレイに表示されます．

⒆ CTRL-B（リターン）

[CTRL] ＋ [B] を入力すると，BASICモードに戻ります．

⒇ CTRL-D（ダンプディスク）

CTRL-Dは，フロッピーディスクの内容をディスプレイに表示させるコマンドです．

CTRL-D〈ドライブ＃〉〔，〈サーフェス＃〉〕，〈トラック＃1〉，〈セクタ＃1〉〔，〈トラック＃2〉，〈セクタ＃2〉〕

〈ドライブ＃〉〔，〈サーフェス＃〉〕，〈トラック＃1〉，〈セクタ＃1〉………表示開始セクタの指定

〔，〈トラック＃2〉，〈セクタ＃2〉〕………表示終了セクタの指定

〈トラック＃2〉，〈セクタ＃2〉を指定しない場合は，

〈表示開始セクタ〉＝〈表示終了セクタ〉

となり，表示開始セクタだけ表示します．

CTRL-Dコマンドを実行中，実行を中止したい場合は，[STOP]または[CTRL]＋[C]を入力します．画面への表示を一時中断したい場合は，[CTRL]＋[S]を入力し，再開するときは，[STOP]および[CTRL]＋[C]以外のキーを入力します．

〈サーフェス＃〉は，画面倍密度のフロッピーディスクユニットを使用しているときに指定します．ただし，モニタでは，両面にまたがってアクセスすることはできません(CTRL-R, CTRL-Wも同じです)．

(21) CTRL-R(リードディスク)

CTRL-Rコマンドは，フロッピーディスクからメモリへデータをロードするコマンドです．

CTRL-R〈ドライブ＃〉〔，〈サーフェス＃〉〕，〈トラック＃〉，〈セクタ＃〉，〈開始アドレス〉，〈終了アドレス〉

このコマンドは，〈開始アドレス〉と〈終了アドレス〉で指定されたメモリの領域に格納するデータを，〈ドライブ＃〉〔，〈サーフェス＃〉〕，〈トラック＃〉，〈セクタ＃〉で指定されたセクタからロードします．ですから，大きなメモリ領域にデータをロードする場合は，複数のセクタ，トラックからデータをロードします．

例

```
h〕^r1,0,9,3,C000,C00f
h〕
```

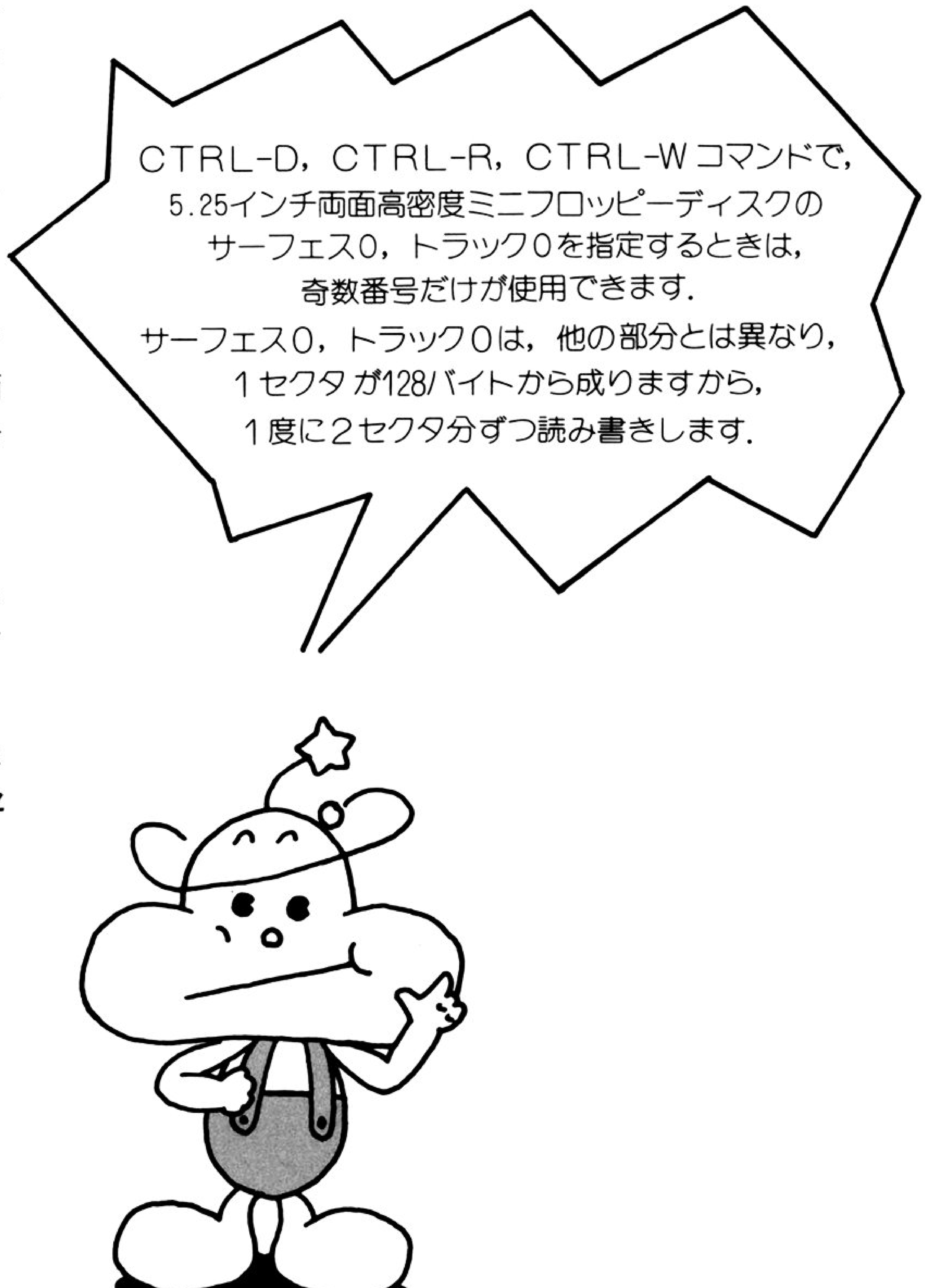
ドライブ1の第9トラックの第3セクタから，16バイトのデータをロードしてきて，C000H番地からC00FH番地に格納します．このとき，メモリ領域のC010H～C0FFH番地の内容は変更されません．

(22) CTRL-W(ライトディスク)

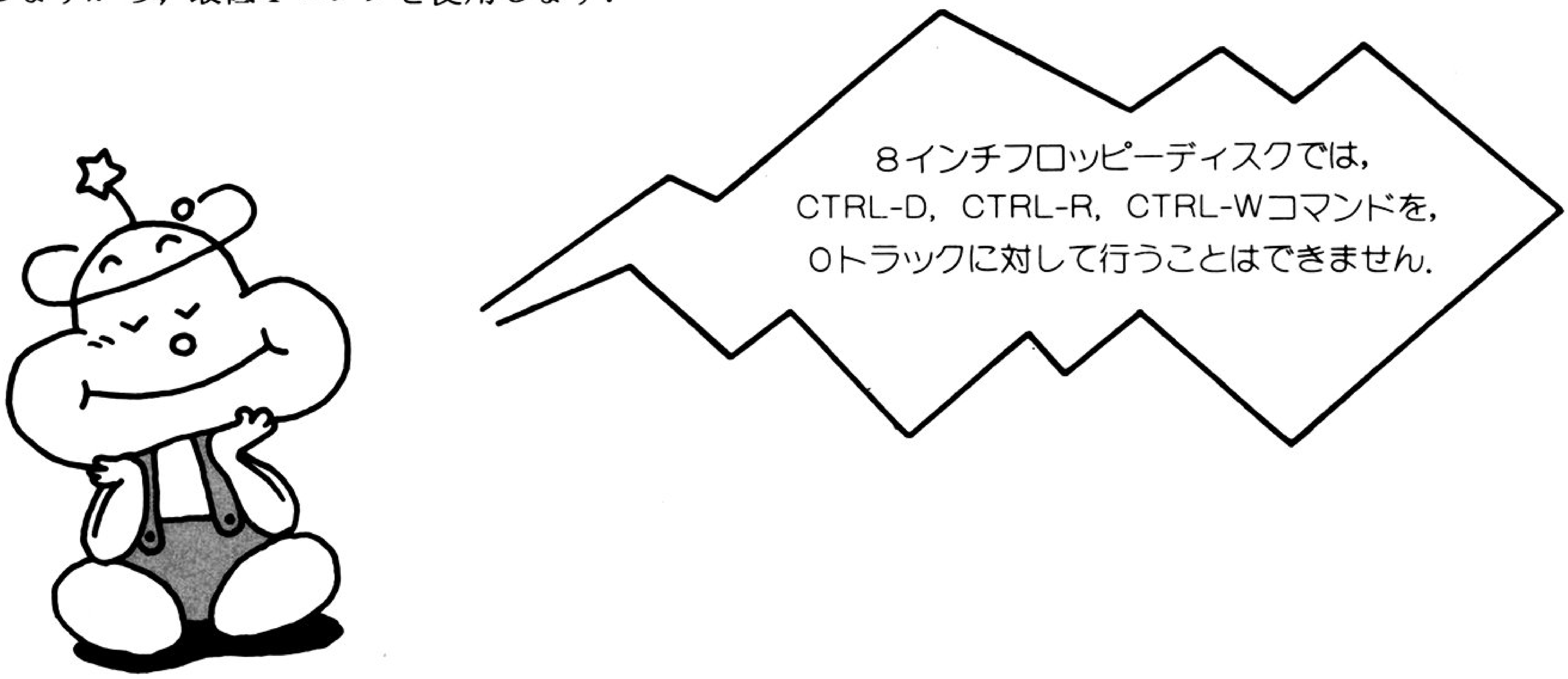
CTRL-Rコマンドと逆の操作を行うコマンドで，メモリの内容をフロッピーディスクへセーブします．

CTRL-W〈ドライブ＃〉〔，〈サーフェス＃〉〕，〈トラック＃〉，〈セクタ＃〉，〈開始アドレス〉，〈終了アドレス〉

〈開始アドレス〉と〈終了アドレス〉は，



セーブするメモリ領域を指定します．このとき，指定する領域は，1セクタ単位(256バイト)である必要はありません．ただし，フロッピーディスクは，セクタ単位で指定しますから，最低1セクタを使用します．



付録1.5 コマンドの種類

機械語モニタのコマンドには，表に示すように22種類のコマンドがあります(命令は大文字でも小文字でも入力できます).

*　CMTインタフェースボードとデータレコーダが必要です.
**　DISK versionでのみ機能します.

コマンド	意味	機能	入力形式	オペレーション
A	アセンブル	入力した1行をアセンブルする.	a〔<格納開始アドレス>〕	インサート……INS デリート………DEL 終了……………STOP, CTRL-C
B	ベース	数値の表現形式を変える.(8進↔16進)	bqまたはbh	――
D	ダンプメモリ	メモリの内容をディスプレイに表示する.	d〔<表示開始アドレス>〕〔,<表示終了アドレス>〕	一時中断………CTRL-S 表示再開………終了キー以外 終了……………STOP, CTRL-C
E	エディトメモリ	スクリーンエディタの機能を用いてメモリの内容を変更する.	e〔<開始アドレス>〕	カーソル移動…カーソル移動キー ロールアップ…ROLL UP ロールダウン…ROLL DOWN 終了……………STOP, ESC
F	フィルメモリ	メモリの内容を定数で埋めていく.	f<開始アドレス>,<終了アドレス>,<定数>	――
G	ゴー	ユーザプログラムを実行する.	g〔<実行開始アドレス>〕〔,<ブレイクポイントアドレス1>〕〔,<ブレイクポイントアドレス2>〕	――
I	インプット	I/Oポートの値を読み込む.	i<ポートアドレス>	――
L	ディスアセンブル	機械語を逆アセンブルする.	l〔<逆アセンブル開始アドレス>〕〔,<逆アセンブル終了アドレス>〕	一時中断………CTRL-S 再開……………終了キー以外 終了……………STOP, CTRL-C
M	ムーブメモリ	ある範囲のメモリの内容を他のアドレスのメモリ領域へ移す.	m<転送元・先頭アドレス>,<転送元・最終アドレス>,<転送先・先頭アドレス>	――
O	アウトプット	I/Oポートへデータを出力する.	o<ポート・アドレス>,<データ>	――

コマンド	意　味	機　能	入力形式	オペレーション
P	プリンタスイッチ	プリンタへの出力をコントロールする。	p	——
R*	リードテープ	カセットテープからデータをロードする。	r〔〈ファイル名〉〕	——
S	セットメモリ	メモリにデータをセットする。	s〔〈開始アドレス〉〕	セット…………スペース，リターン アドレス後退……CTRL-B 終了……………リターン
TM	テストメモリ	メモリをテストする。	tm	——
V*	ベリファイテープ	カセットテープの内容と，メモリの内容を比較する。	v〔〈ファイル名〉〕	——
W*	ライトテープ	メモリの内容をカセットテープにセーブする。	w〔〈レジスタ名〉〕， 〈セーブ開始アドレス〉 〈セーブ終了アドレス〉	——
X	イグザミンレジスタ	CPUのレジスタの値を調べ，変更する。	x〔〈レジスタ名〉〕	次レジスタ……スペース 終了……………リターン
HELP または CTRL-A	ヘルプ	コマンドとそのパラメータの形式をディスプレイに表示する。	CTRL-A または HELP	——
CTRL-B	リターン	BASIC モードへ復帰する。	CTRL-B	——
CTRL-D**	ダンプディスク	フロッピーディスクの内容をディスプレイに表示する。	CTRL-D 〈ドライブ#〉〔,〈サーフェス#〉〕, 〈トラック#1〉,〈セクタ#1〉 〔,〈トラック#2〉,〈セクタ#2〉〕	——
CTRL-R**	リードディスク	フロッピーディスクから，データをロードする。	CTRL-R 〈ドライブ#〉〔,〈サーフェス#〉〕, 〈トラック#〉,〈セクタ#〉, 〈開始アドレス〉,〈終了アドレス〉	——
CTRL-W**	ライトディスク	フロッピーディスクへデータをセーブする。	CTRL-W 〈ドライブ#〉〔,〈サーフェス#〉〕, 〈トラック#〉,〈セクタ#〉, 〈開始アドレス〉,〈終了アドレス〉	——

コンピュータどうしをつなごうとする人のために

付録2. ターミナルモード

PC-8801MKⅡMRは，ターミナル用のインタフェースとして，RS-232Cの規格のインタフェースを持っています．

付録2では，PC-8801MKⅡMRをターミナル（端末）として使う方法を説明します．ここで説明するターミナルモードは，PC-8801MKⅡMRに内蔵されている簡単ターミナルプログラムを使う場合です．このターミナルプログラムは，日本語を扱うことができません．

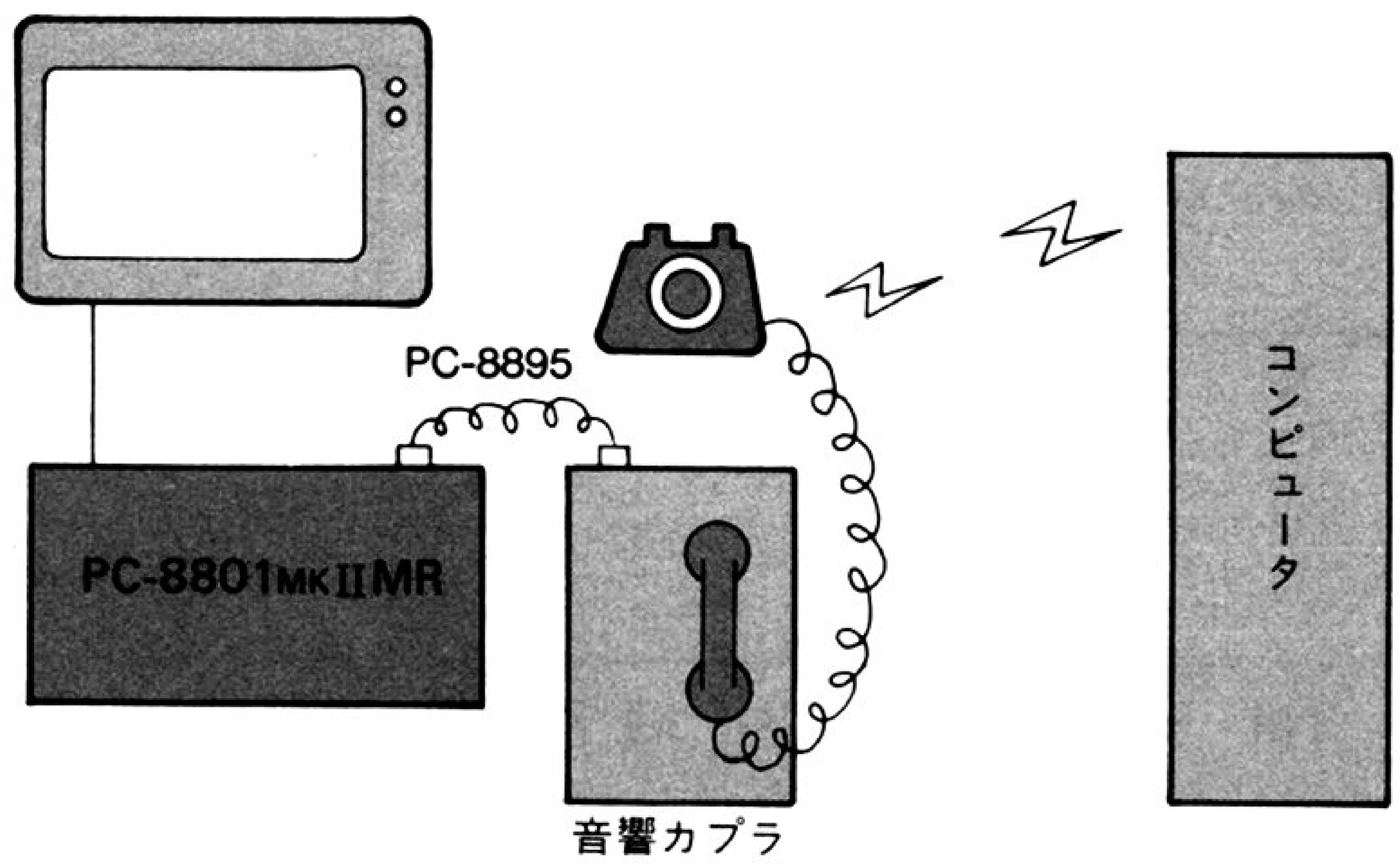
モデムを接続して，ターミナルモードを使用する場合には，以下の制限があります．

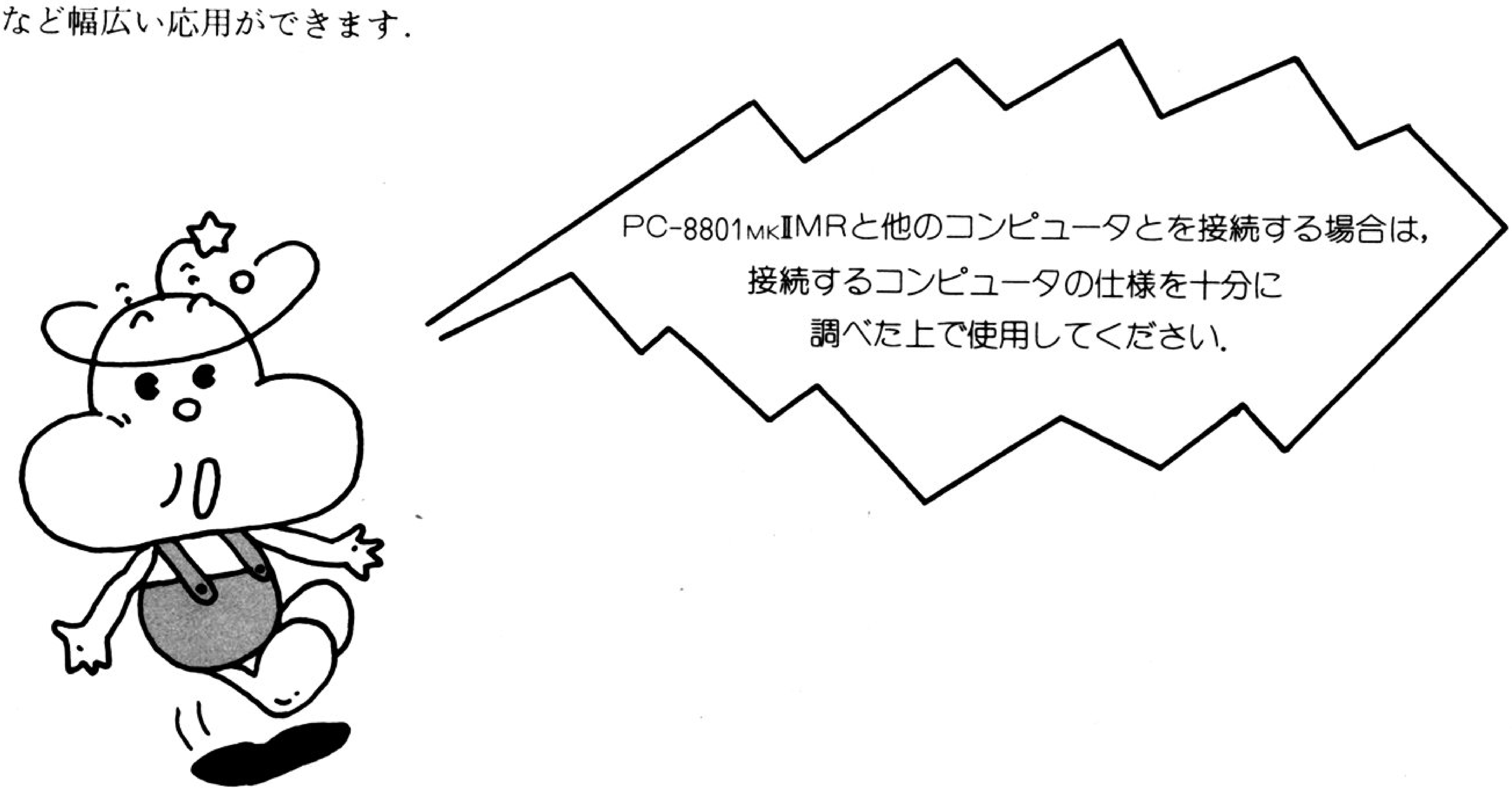
(1) 調歩同期モード（非同期モード）のモデムだけが使用可能です．同期モードのモデムは使用できません．

(2) 全二重モデム（送信，受信それぞれ用の回線を使う形式）だけが使用可能です．

制御線を使って通信する半二重モデムは使用できません．

PC-8801MKⅡMRをターミナルとして使うことによって，遠く離れた中央のコンピュータ（ホストコンピュータ）から送られてきたデータを受け取ったり，電話回線を通して各種のコンピュータサービスを受けたりなど幅広い応用ができます．





付録2.1 ターミナルモードへの切り替え

PC-8801mkⅡMRを他のコンピュータ(ホストコンピュータ)のターミナル(端末)として使用するには，ターミナルモードに切り替えて使用するのが便利です．

PC-8801mkⅡMRをターミナルモードにするには，次の3つの方法があります．

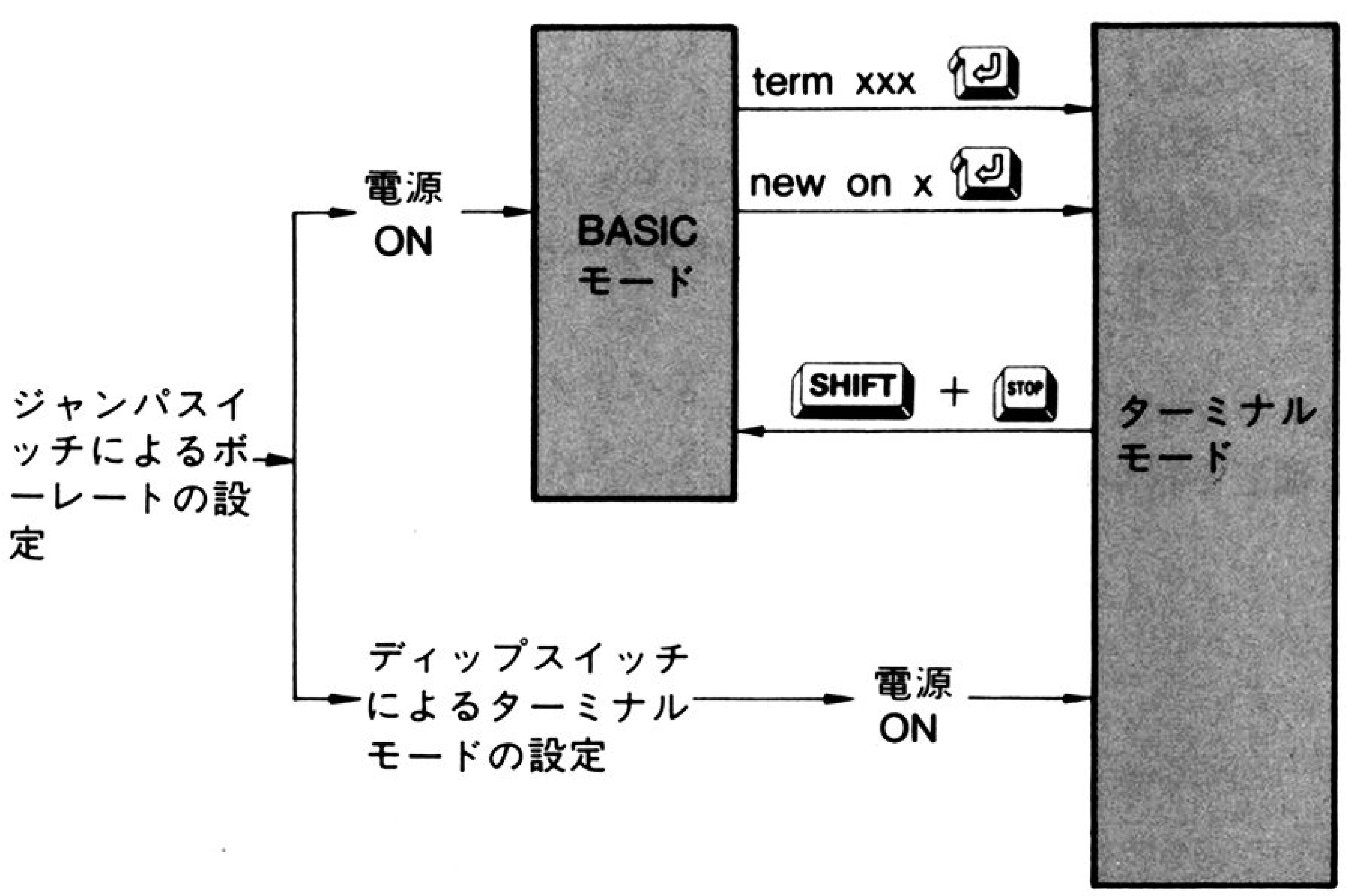
(1) BASICモードからTERM文によって切り替える．

(2) BASICモードからNEW ON文によって切り替える．

(3) あらかじめPC-8801mkⅡMR本体前部のディップスイッチによってターミナルモードに設定しておき，電源ONと同時にターミナルモードにする．

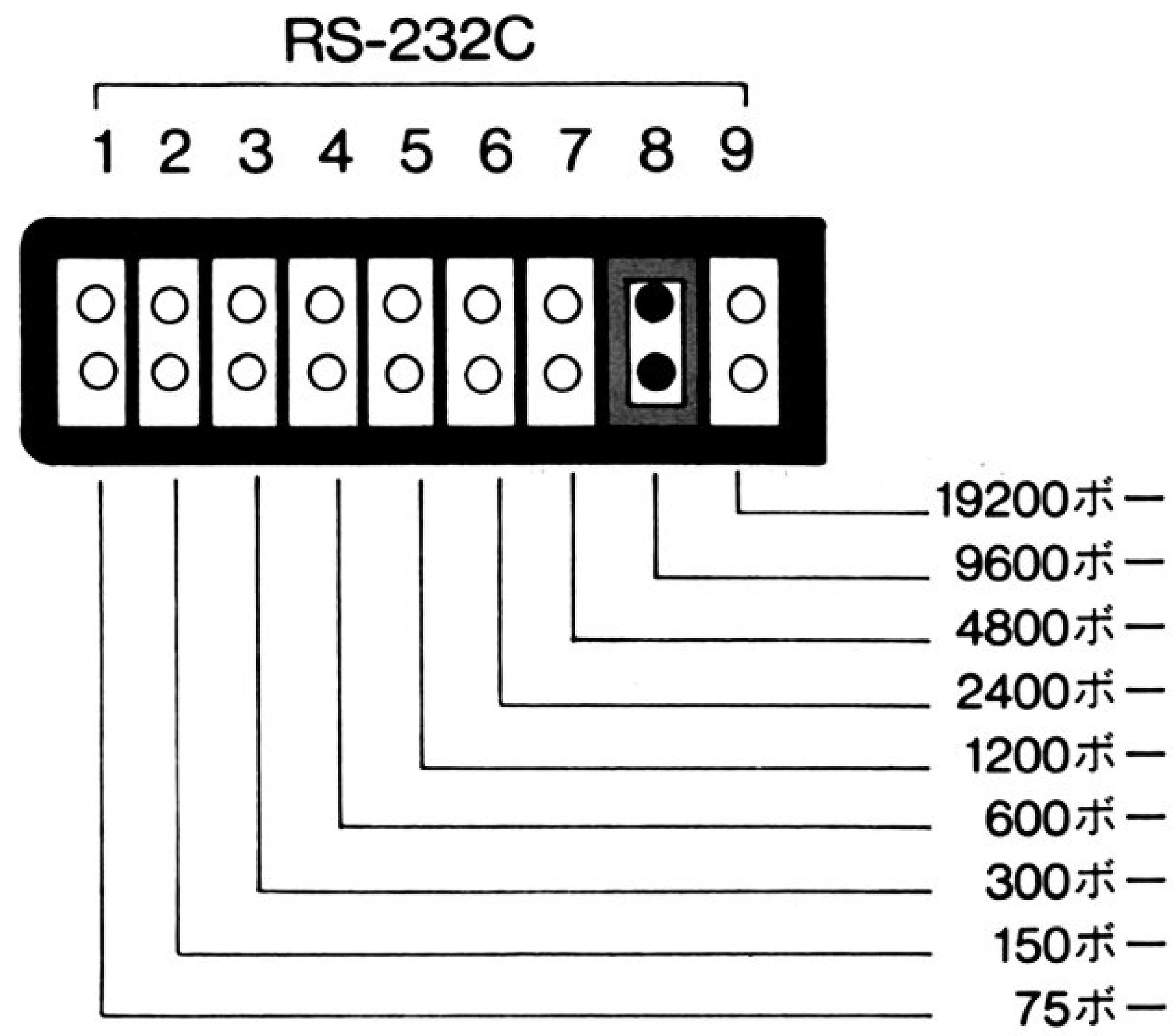
(PC-8801mkⅡMRをターミナルモードでしか使用しない場合は，(3)の方法が便利です)．

また，PC-8801mkⅡMRがターミナルモードになっているときに，SHIFT を押しながら STOP を押すと，BASICモードに切り替わります．



1　ボーレートの設定

PC-8801MKⅡMRをターミナルで使用するには，あらかじめ本体前部のジャンパスイッチによってボーレート(伝送速度)を切り替えておきます．出荷時は9600ボーにセットされていますから，ホストコンピュータの仕様に合わせて設定してください．



2　TERM文を使用してBASICモードからターミナルモードに切り替える方法

BASICをスタートさせ，次のTERM文を入力します．各パラメータは，大文字を使用してください．

term "com:〈パリティ〉〈ビット長〉〈ストップビット〉〈Xパラメータ〉〈Sパラメータ〉"〔,〔〈通信モード〉〕〔,〈スタック長〉〕〕

Xパラメータを有効にすると，データがPC-8801MKⅡMRのデータ転送用バッファの4分の3までデータが入ると，自動的にホストコンピュータにCTRL-Sコードを出力してデータ転送の一時停止を要求し，4分の1になるとCTRL-Qを出力して転送の再開を要求します．

この機能は，ホストコンピュータがXパラメータを処理する機能を持つ場合に限って有効になります．

〈パリティ〉：パリティチェックの指示を行う．

偶数パリティ(even parity) ……………………………… E
奇数パリティ(odd parity)………………………………… O
パリティチェックなし(non parity)…………………… N

〈ビット長〉：データのビット長を表す．

7ビット………………… 7
8ビット………………… 8

〈ストップビット〉：ストップビットの長さを表す．

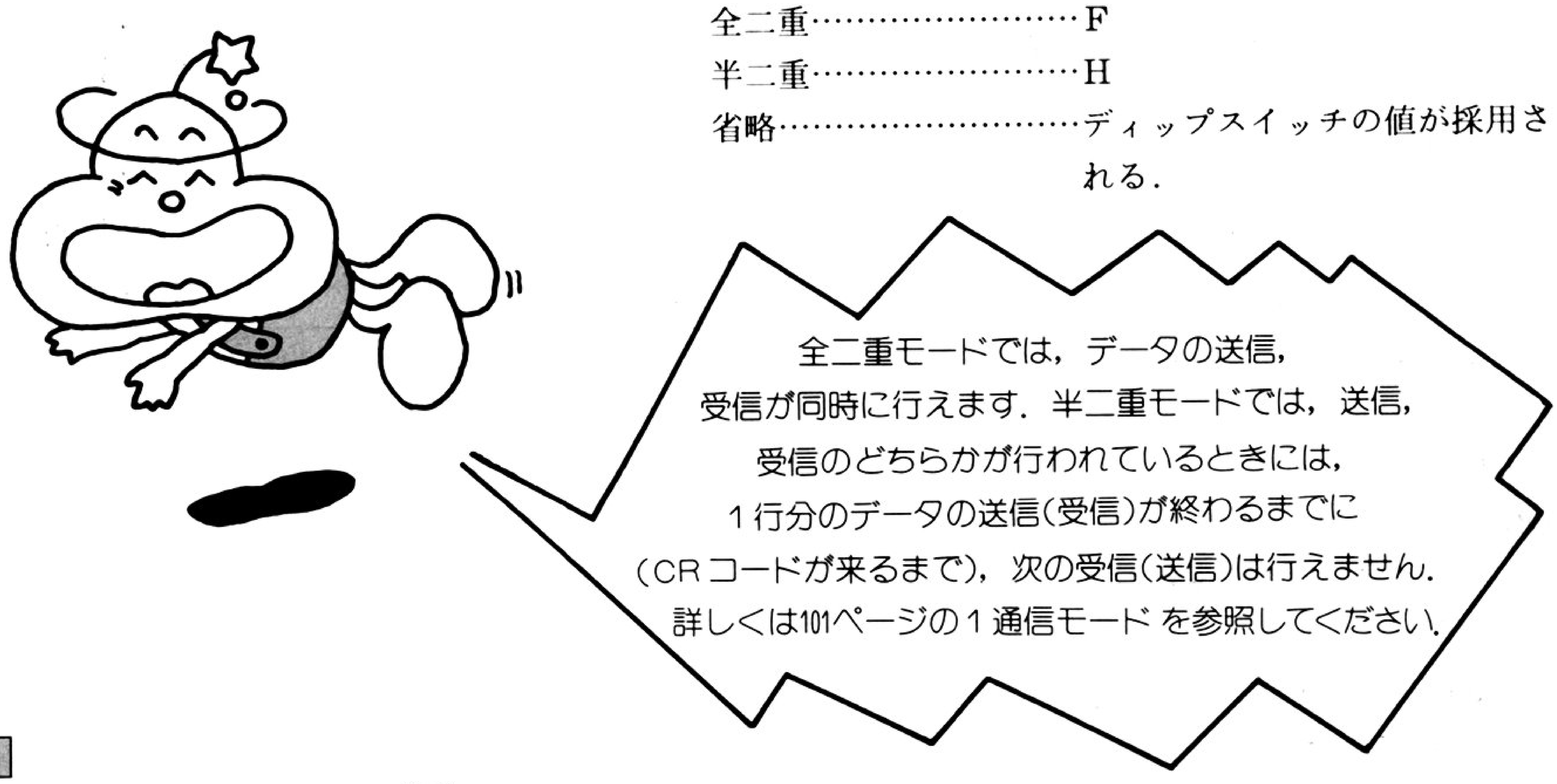
1ビット………………… 1
1.5ビット ……………… 2
2ビット………………… 3

〈Xパラメータ〉：Xパラメータの有効，無効の切り替えを行う．

Xパラメータ無効………N
Xパラメータ有効………X
Xパラメータ省略………ディップスイッチの値が採用される．



Sパラメータを有効にすると，7ビットモードでカナを送受信することができます．つまり，SOコードがくるとそれ以後のデータをカナとみなし，SIコードがくるとそれ以後のデータを英数字とみなします．



例

term□"com:E72X",H⏎

偶数パリティ，データ長7ビット，ストップビット1.5ビット，Xパラメータ有効，半二重モードのターミナルモードになります．

TERM文のパラメータは，次のように省略することができます．

term□"com:"⏎
term□"□"⏎

この場合は，各パラメータともディップスイッチの値，または標準値(スタック長)が設定されます．

ターミナルモードになっているPC-8801mkⅡMRからホストコンピュータにBreak信号を送る場合は，STOPを入力します．

〈Sパラメータ〉：Sパラメータの有効，無効の切り替えを行う．

Sパラメータ有効………S
Sパラメータ無効………N
Sパラメータ省略………ディップスイッチの値が採用される．

〈通信モード〉：全二重，半二重の指定を行う．

全二重……………………F
半二重……………………H
省略………………………ディップスイッチの値が採用される．

〈スタック長〉：ターミナルモードでも，ホストコンピュータから送られてきたBASICのコマンドを実行することができますが，そのとき，使用する演算スタックの大きさを指定します．省略すると1024バイトが設定されます．

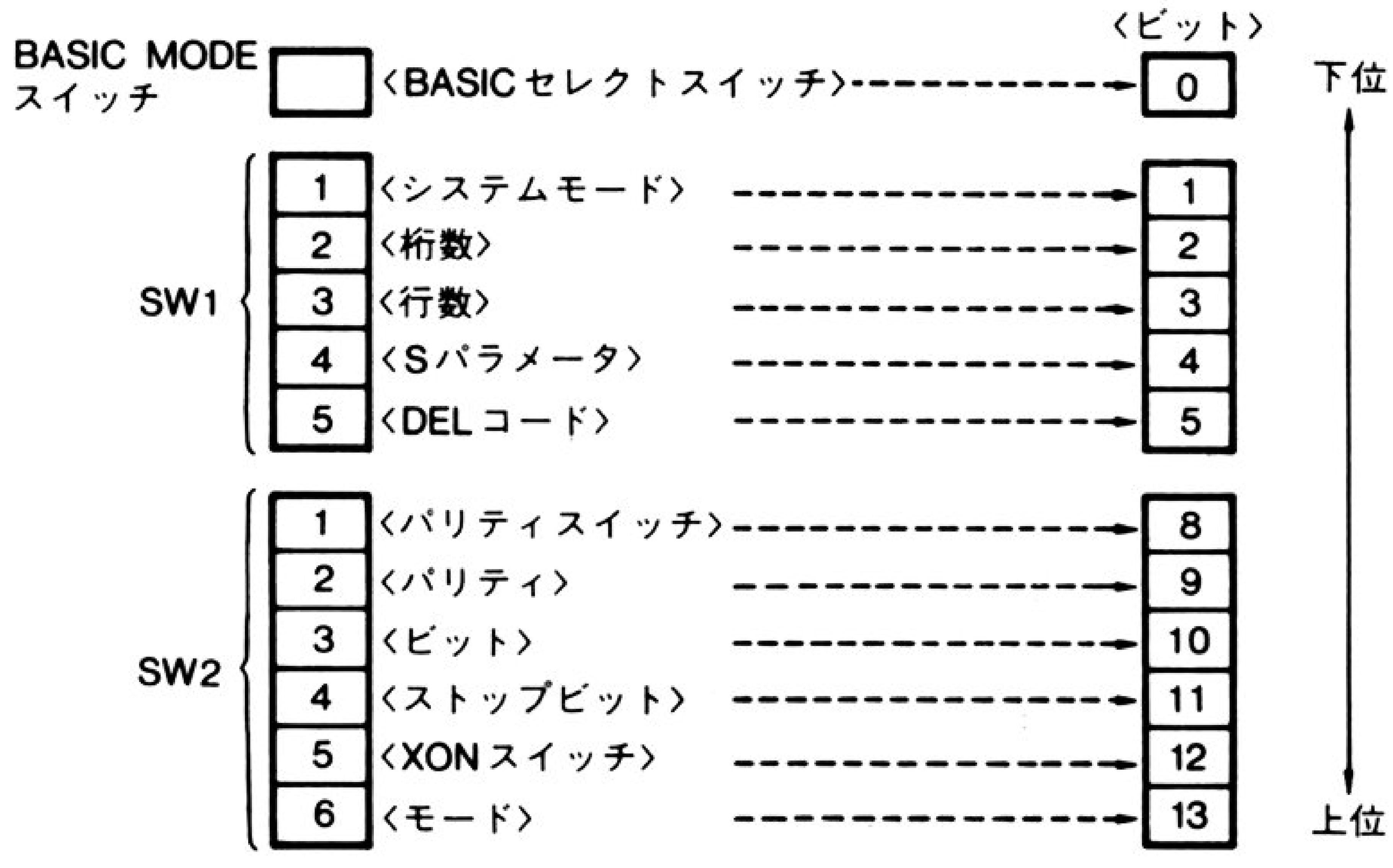
3　NEW ON文を使用してBASICモードからターミナルモードに切り替える方法

BASICをスタートさせ，次のNEW ON文を入力します．

new on 〈式〉

NEW ON文はBASIC MODEスイッチ，ディップスイッチによる各モードの設定を，ソフトウェアによって行うためのもので，〈式〉の値はBASIC MODEスイッチ，ディップスイッチの1個が1ビットに対応した(ONのとき1，OFFのとき0)16ビット表現の値を指定します．

NEW ON文によって指定できるのは，次のBASIC MODEスイッチおよびディップスイッチです．その詳細については，次節およびBASICリファレンスマニュアルを参照してください．



(注)　0, 6, 7, 14, 15ビットは意味を持たないので0にしてください．

BASIC MODEスイッチおよび
ディップスイッチとの対応関係

例

new on 4866 ⏎

とすれば，

term "com:E71X,H ⏎

と同じターミナルモードになります(ただし，この場合数値はディップスイッチによって設定されている画面の桁数，行数によって異なってきます)．

4　ディップスイッチでターミナルモードに切り替える方法

(1)　周辺機器，およびPC-8801MKⅡFRの電源をOFFにします．

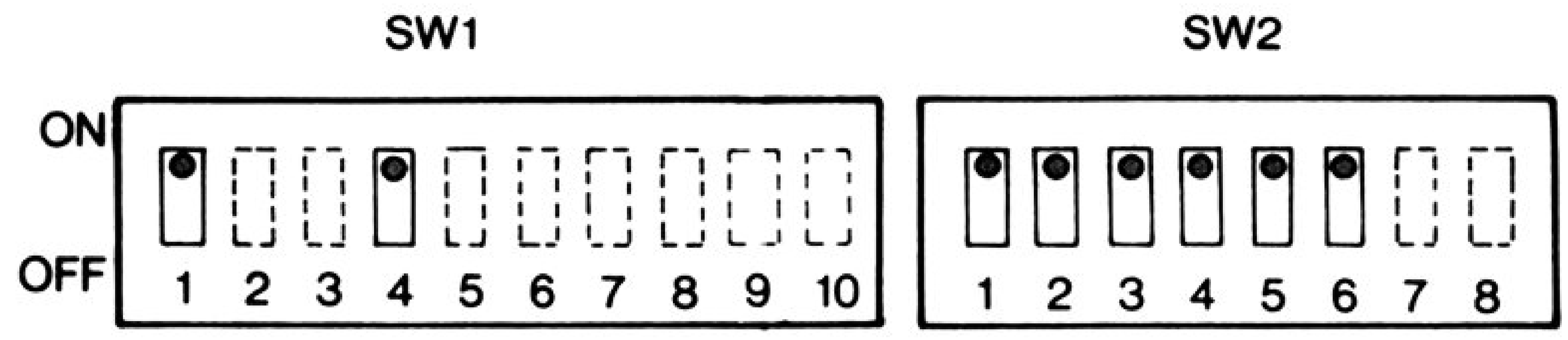
(2)　PC-8801MKⅡMR本体前面のBASIC MODEスイッチをN88V1またはN88V2にし，ディップスイッチSW1の1をON(上向き)にします．

(3)　他のディップスイッチによって，各種モードを設定します．

(4)　周辺機器およびPC-8801MKⅡMRの電源をONにします．これで最初からターミナルモードがスタートします．

ディップスイッチ		ON(上向き)	OFF(下向き)
SW1	4	Sパラメータ有効	Sパラメータ無効
	5	DELコードを処理する	DELコードを無視する
SW2	1	パリティチェックあり	パリティチェックなし
	2	偶数パリティ	奇数パリティ
	3	データ長が8ビット	データ長が7ビット
	4	ストップビットが2ビット	ストップビットが1ビット
	5	Xパラメータ有効	Xパラメータ無効
	6	半二重モード	全二重モード

例 偶数パリティ，8ビット長，ストップビット長が2ビット，Xパラメータ有効，Sパラメータ有効，半二重モードに設定する場合(term "com:E83XS",Hと同じです)．



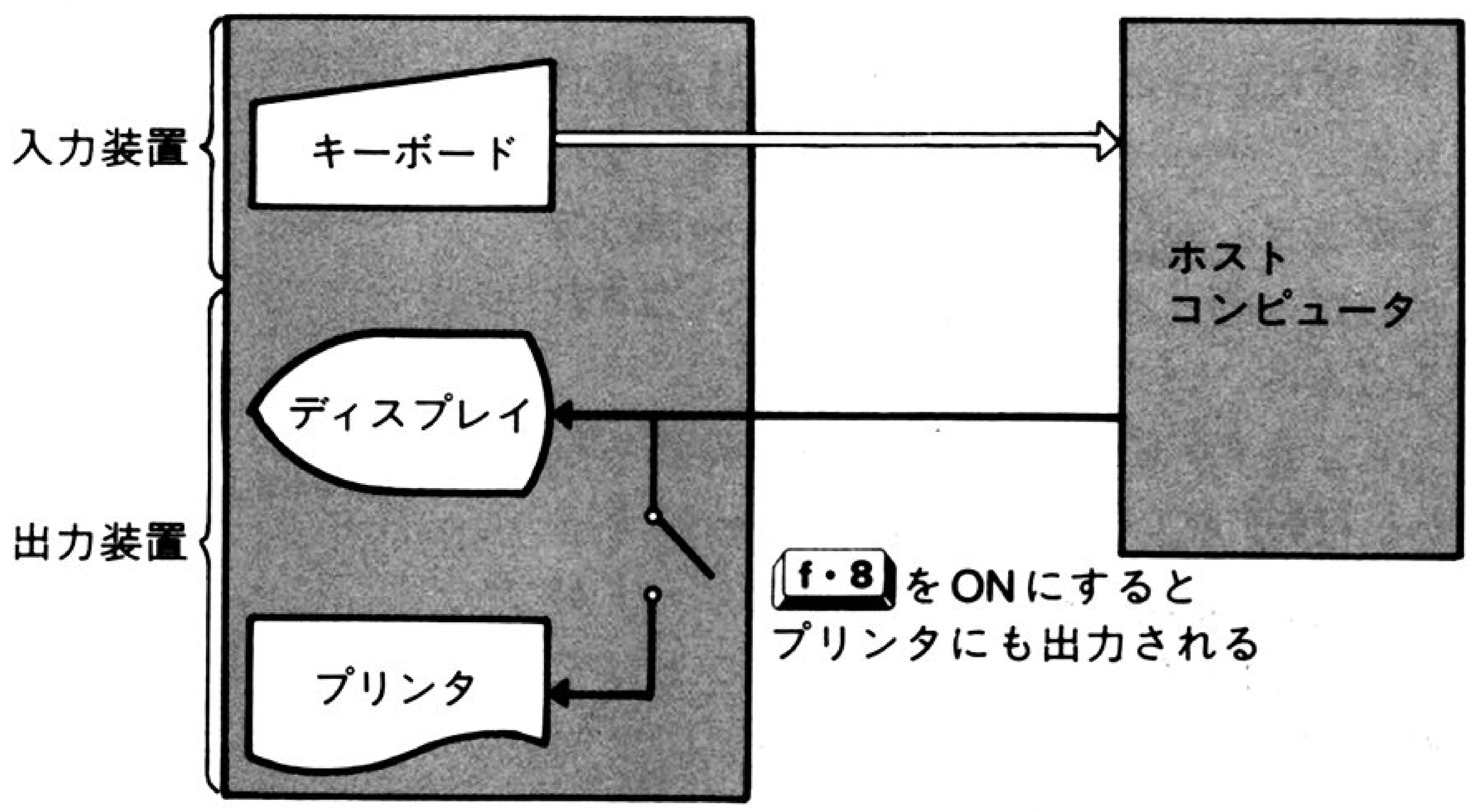
付録2.2 ターミナルモードの機能

1 通信モード

PC-8801MKⅡMRをターミナルモードにしてデータの送受信を行うとき，通信モードは，全二重モードまたは半二重モードで行います．

(1) 全二重モード

全二重モードでは，データの送受信を同時に行うことが可能です．

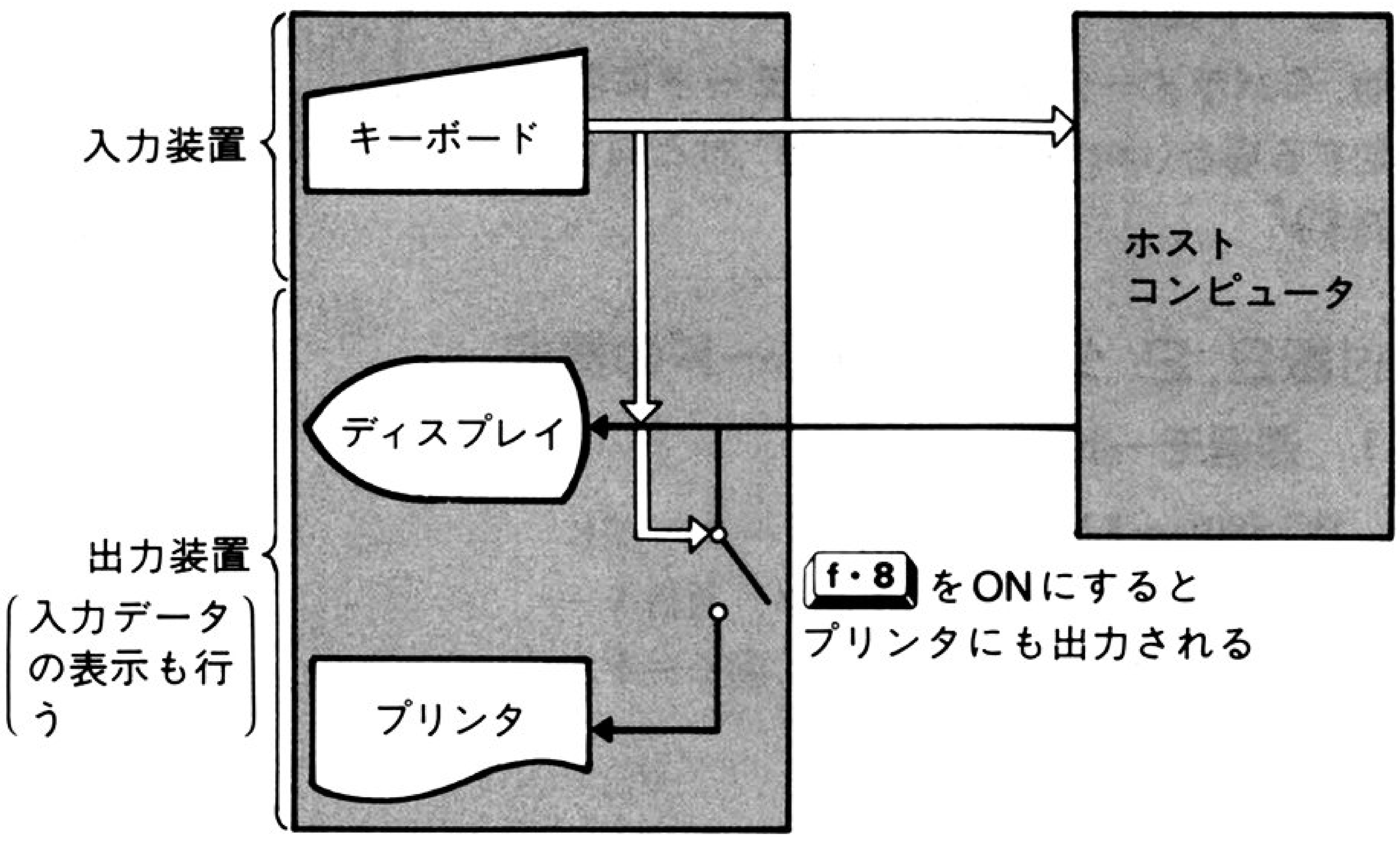


PC-8801MKⅡMRシステム

キーボードから入力されたデータは，ただちに回線に送出されますが，PC-8801MKⅡMRに接続されているディスプレイ，プリンタへは表示・出力されません．

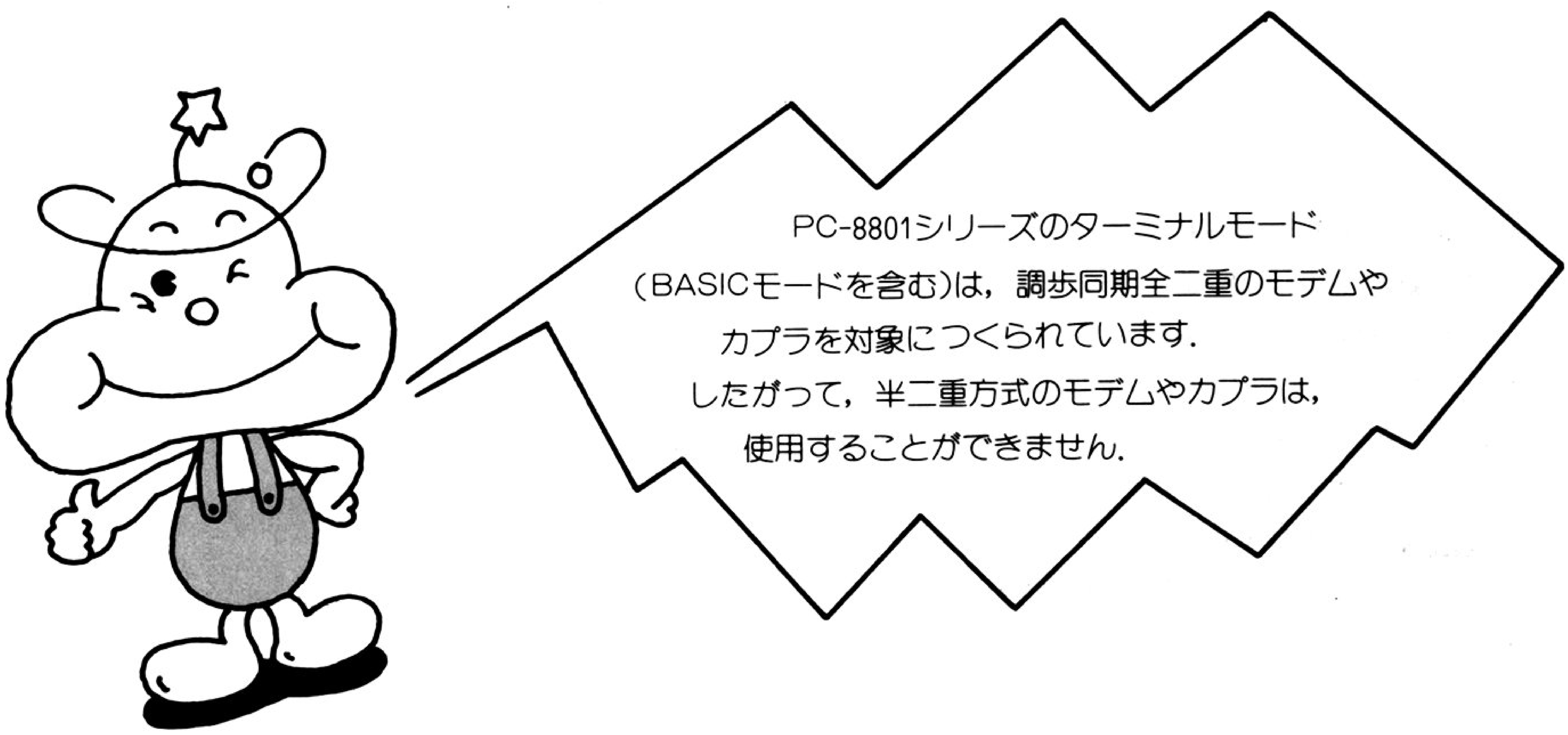
受信データは自分のディスプレイに表示されますが，そのとき，ファンクションキー f・8 がON(LPT enable)であれば，プリンタにも出力され，OFF(LPT disable)であれば出力されません．

(2) 半二重モード



キーボードから入力れたデータは，PC-8801mkⅡMRに接続されているディスプレイにいったん表示され，⏎が押されると，データを相手に送出します．またこのとき，f・8がON(LPT enable)であれば，キーボードからデータが入力されるごとに，プリンタに出力します．

半二重モードでは，送信，受信のどちらかが行われているときには，1行分のデータの送信(受信)が終わるまで(CRコードがくるまで)次の送信(受信)を行うことはできません．



2 ファンクションキー

ターミナルモードに切り替えると，ファンクションキー f･1 ～ f･5 は，ターミナルモードになる前にBASICモードで定義されたものになりますが，ファンクションキーの f･6 ～ f･10 は次のように定義されます．

f･6 literal

このキーを入力すると，コントロールコードが実行されず，文字として表示されます．ただし，リターンとラインフィードは実行されます．

f･7 half / full

全二重，半二重の切り替えを行います．初期状態は，パラメータまたはディップスイッチで設定されます．

f･8 LPT enable

画面への出力をプリンタへも出力します．もう一度入力すると出力を停止します．初期状態は出力しないモードに設定されています．

f･9 copy buffer

画面以外のバッファの内容をプリンタへ出力します．

f･10 LPT feed

プリンタに，1行フィードするコードを出力します．

3 スペシャルエスケープシーケンス

ターミナルモードに切り替えると，BASICのモードから抜け出てしまいますが，スペシャルエスケープシーケンスを利用すれば，BASICの命令を実行できます．

スペシャルエスケープシーケンスは，ホストコンピュータから次に示すようなエスケープコードを送ることによって，ターミナルモードのPC-8801MKⅡMRに接続されているディスプレイの画面の表示桁数を変えたり，文字の色を変えることができます．また，BASICの命令を実行するだけでなく，プリンタへのエコーバックの切り替えなども，この機能を用いれば簡単に行えます．右に個別の機能を説明します．

(1) ESC@ ：ディスプレイ画面とプリンタへの出力(エコーバック)を開始する．

(2) ESC A ：ディスプレイ画面とプリンタへの出力(エコーバック)を解除する．

(3) ESC" ：PC-8801MKⅡMRのキー入力を有効にする．

(4) ESC# ：PC-8801MKⅡMRのキー入力を無効にする．

(5) ESC! ：ホストコンピュータへIDコードを出力する．

(6) ESC ＊ ：現在スクロールされている範囲をクリアする．

(7) ESC Y ：現在のカーソル位置からスクロールされている最下位行までををクリアする．

(8) ESC␣ ：ターミナルのイニシャライズを行う(TERMコマンドを実行した直後と同じ状態にする)．

例

カーソルの位置を(3, 8)に移動させるには,

3 → 23H = '#'

8 → 28H = 'c'

ですから,

ESC = #C

とします.

4 リモートBASICプロトコル

前節スペシャルエスケープシーケンスで説明した機能の中で,

(11) ESC>BASIC STATEMENT

(12) ESC . BASIC STATEMENT

(13) ESC<BASIC STATEMENT

の3つを,リモートBASICプロトコル(remote BASIC protocol)といいます.

これによって,ホストコンピュータから,ターミナルとなっているPC-8801MKⅡ MRのBASICの機能が利用できるようになります.ただし,この機能にはいくつかの制約があります.

(i) 使用できないBASICステートメント

リモートBASICプロトコルでは,メモリ内にプログラムを作成したり,メモリ内のプログラムの変更,削除は行えません.また,メモリ内の変数エリアを直接扱うステートメント(たとえば,fre(0),clear, eraseなど)も実行できません.さらに,同じバッファを使って入出力を行うファイルをOPENさせることはできません(com:, cas:).

(ii) 実行結果の表示

PRINT文やPRINT USING文など実行結果を画面に表示させる命令では,表示をターミナル側に行うか,ホストコンピュータ側で行うかを選択できます.すなわち,エスケープコード(ESC)の後の符号(>,

(9) ESC =〈X-pos〉〈Y-pos〉

: カーソル位置を指定する.(20Hのバイアスがかかる.)

(10) ESC T : 現在カーソルがある行を1行消去する.

(11) ESC>BASIC STATEMENT

: BASICステートメントを実行する.
BASICの実行結果は,ターミナルとなっているPC-8801MKⅡ MR側に表示される.

(12) ESC . BASIC STATEMENT

: BASICステートメントを実行する.
ただし,実行結果は表示されない.

(13) ESC<BASIC STATEMENT

: 実行結果の表示はホストコンピュータ側だけ.

<)によって指定できるわけです．しかし，グラフィック命令はこの符号の指定にかかわらず，必ずターミナルとなっているPC-8801MKⅡMRに接続されているディスプレイに実行結果を表示します．

(iii) 使用変数の制限

リモートBASICプロトコルでは，使用できる変数エリアが定まっています．また，一度使用するとそのエリアは二度と別の変数には使えませんから，多くの変数に値を代入していくと，途中で代入できなくなってしまいます．リモートBASICプロトコルで使用する変数は，必要最小限にとどめてください．

例

● ターミナルにグラフを表示させる．

次のようにホストコンピュータからコードを送ると，ターミナルに鮮やかなグラフが簡単に表示できます．

```
ESC>screen0,0
ESC>circle(100,50),100,7
ESC>a=100:b=100:c=200:d=
50
ESC>line(100,50)-(a,b),7
ESC>line(100,50)-(c,d),7
ESC>paint(120,70),1,7
```

● BASICからターミナルモードに入った場合

次のようにすると，ターミナルになっているPC-8801MKⅡMRに接続されているフロッピーディスクに，データをセーブできます．

```
ESC>open "2:TEST1 "for output
as #1
ESC>print#1,100
```

付録3. シンセサイザICの構造

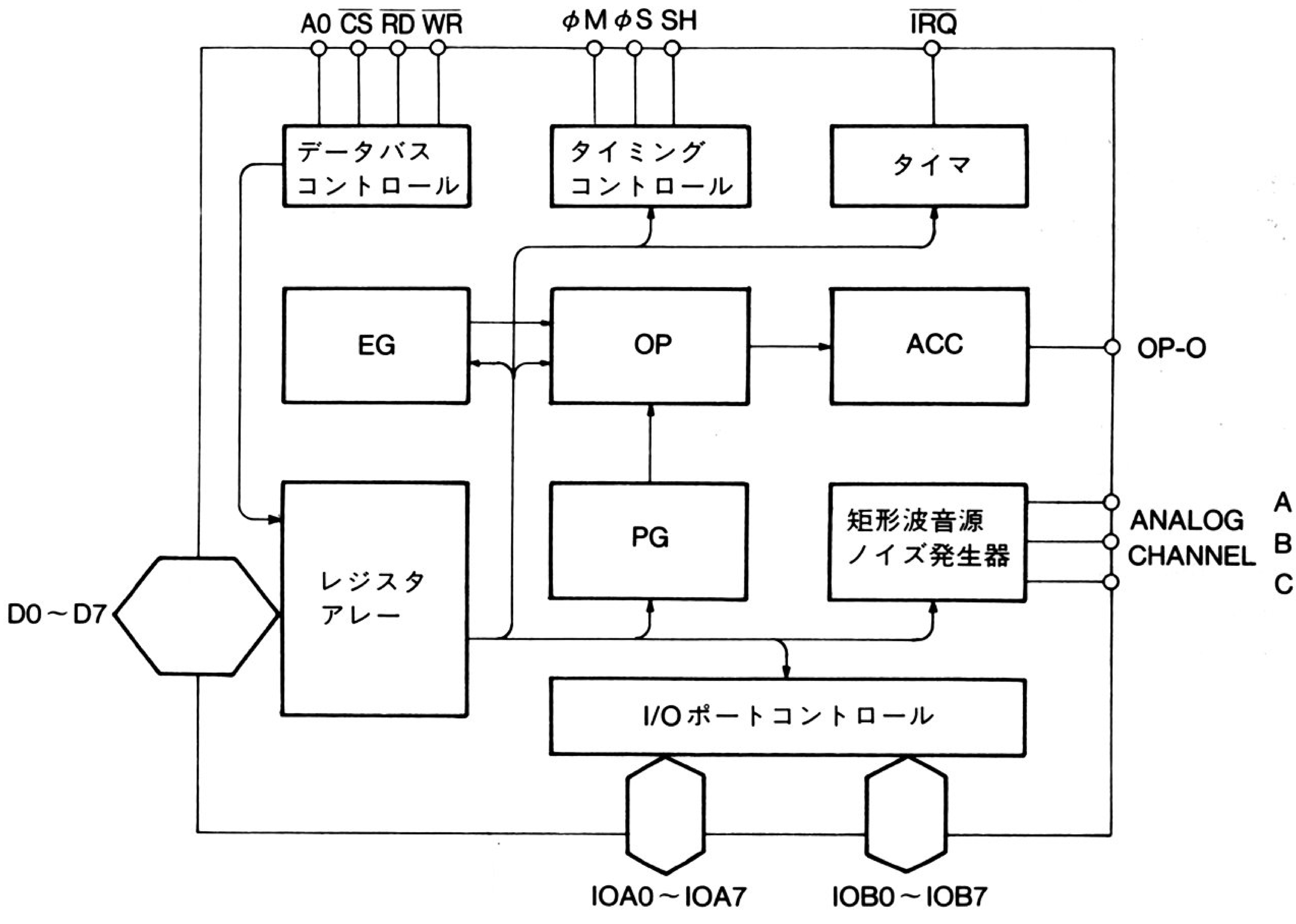
この節はBASICだけでシンセサイザIC を使用するユーザは読む必要がありません.

OPNはFM音源, SSG音源, 2つのタイマAとBから構成されます. ここでは, FM音源, SSG音源を中心に説明します.

(付録3は日本楽器製造㈱ YM-2203アプリケーションマニュアルをもとにしています).

付録3.1 FM音源部

FM音源部は, FM音源をコントロールするデータのレジスタ(REG)・フェイズジェネレータ(PG)・エンベロープジェネレータ(EG)・FMの演算を行うオペレータ(OP)および各スロットの和を求めるアキュムレータ(ACC)から成り立っています.



ブロックダイアグラム

(a)　レジスタ(REG)

レジスタには，FM音源に対するレジスタと，SSG音源に対するレジスタの2種があります，

FM音源部のレジスタのエリアは，表1のアドレスマップで示されるように，21HからB2Hまでの146バイトのなかで割りふられています．ここでいうアドレスとは，OPN内で各レジスタに割りふられたサブアドレスであり，このアドレスのアサインはデータビットD0〜D7で行います．したがって，あるレジスタにデータを書き込む場合には，データに先立ってサブアドレスのデータを送り，次にFMデータを送ることになります．サブアドレスについては，いったんあるアドレスを書き込むと，次にアドレスを書き換えるまで，そのアドレスは有効であり，同一のアドレスを何度もアクセスする場合は，アドレスの書き込みは最初に設定するだけで，その後については不要です．アドレスとFMデータの切り替えは，A0入力で行われ，A0＝"0"すなわちLowレベルのときは，データビットD0〜D7上のデータはサブアドレスを意味し，A0＝"1"すなわちHighレベルのときはFMデータとなります．この手法は，FM音源部のデータが書き込みだけしかできないのに対して，SSG部ではレジスタの内容を読むこともできるという点を除けば，SSG部において同様にレジスタのアクセスを行うことができます．

なお，全レジスタとも初期設定(イニシャルクリア：$\overline{\mathrm{IC}}$端子＝"0")のときに，オール0にされます．

表1　OPNアドレスマップ1

※ WRITE DATA ADDRESS	DATA	COMMENT
21H	TEST	LSIのTEST-DATA
24H	TIMER-A	TIMER-Aの上位8bits
25H	TIMER A	TIMER-Aの下位2bits
26H	TIMER-B	TIMER-BのDATA
27H	MODE / RESET B A / ENABLE B A / LOAD B A	TIMER-A/BのControlおよび3CHのMode
28H	SLOT / CH	Key-on/off
2DH		プリスケーラをSet
2EH		1/3, 1/6分周の選択
2FH		分周器を1/2分周にセット
30H〜3EH	DT / MULTI	Detune/Multiple (33H, 37H, 3BHのAddressは無し)
40H〜4EH	TL	Total Level (43H, 47H, 4BHのAddressは無し)
50H〜5EH	KS / AR	Key Scale / Attack Rate (53H, 57H, 5BHのAddressは無し)
60H〜6EH	DR	Decay Rate (63H, 67H, 6BHのAddressは無し)
70H〜7EH	SR	Sustain Rate (73H, 77H, 7BHのAddressは無し)
80H〜8EH	SL / RR	Sustain Level / Release Rate (83H, 87H, 8BHのAddressは無し)
90H〜9EH	SSG-EG	SSG-Type Envelop Control (93H, 97H, 9BHのAddressは無し)
A0H A1H A2H	F-Num. 1	F-Numbers/BLOCK
A4H A5H A6H	BLOCK / F-Num. 2	
A8H A9H AAH	3CH＊F-Num. 1	3CH-3slot F-Numbers/BLOCK
ACH ADH AEH	3CH＊BLOCK / 3CH＊F-Num. 2	
B0H B1H B2H	FB / CONNECT	Self-Feedback/Connection

表2　OPNアドレスマップ2

※READ/WRITE DATA

ADDRESS					COMMENT
00H	Fine Tune				Channel-A Tone Period
01H		Coarse Tune			
02H	Fine Tune				Channel-B Tone Period
03H		Coarse Tune			
04H	Fine Tune				Channel-C Tone Period
05H		Coarse Tune			
06H		Period Control			Noise Period
07H	IN/OUT IOB IOA	/Noise	/Tone		/ENABLE
08H		M	Level		Channel-A Amplitude
09H		M	Level		Channel-B Amplitude
0AH		M	Level		Channel-C Amplitude
0BH	Fine Tune				Envelop Period
0CH	Coarse Tune				
0DH		C	ATT	ALT / HLD	Envelop Shape, Cycle
0EH	I/O Port-A				I/O Port Date
0FH	I/O Port-B				

※READ DATA

ADDRESS				COMMENT
00H～0FH	BUSY		FLAG B A	Status

(a—1)　TEST : ADDRESS[21H]

このアドレスは，OPNをテストするために設けられたのであり，常に"0"レベルにしておかなければなりません．

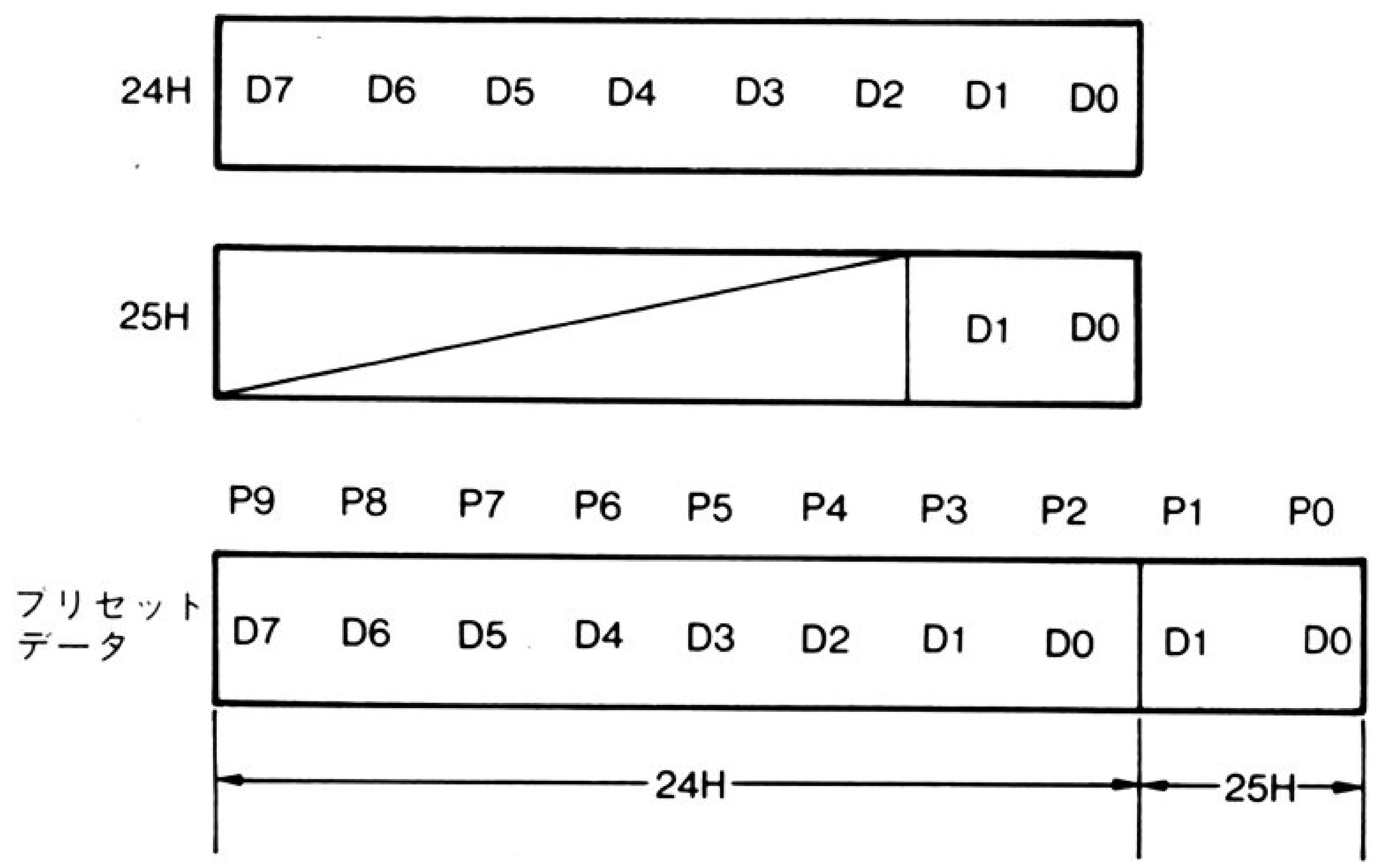
(a—2)　TIMER

タイマは2個持っており，1つは10ビットプリセッタブルタイマ(TIMER-A)で，もう一つは8ビットプリセッタブルタイマ(TIMER-B)です．各タイマは始動・停止およびフラグの制御が可能です．

(Ⅰ) TIMER-A : ADDRESS[24H]・[25H]

タイマAは，アドレス24Hと25Hの10ビットからなり，この10ビットの値をプリセット値としてカウンタを動かします．そして，カウンタがオーバーフローを生じたときにタイマAのフラグを立て，同時にプリセット値をロードします．

タイマAは，通常のタイマ機能以外にCSMのコントロールとしても働きます．この場合は，オーバーフローが生じたときのみチャネル3の各スロットをONにして，チャネル3の全スロットの音を出します．これにより，複合正弦波合成による音声合成が可能になります．



T　0VA(ms)＝12＊(1024－NA)/f FM(kHz)

NA： P9＊2^9＋P8＊2^8＋P7＊2^7＋…＋P1＊2^1＋P0

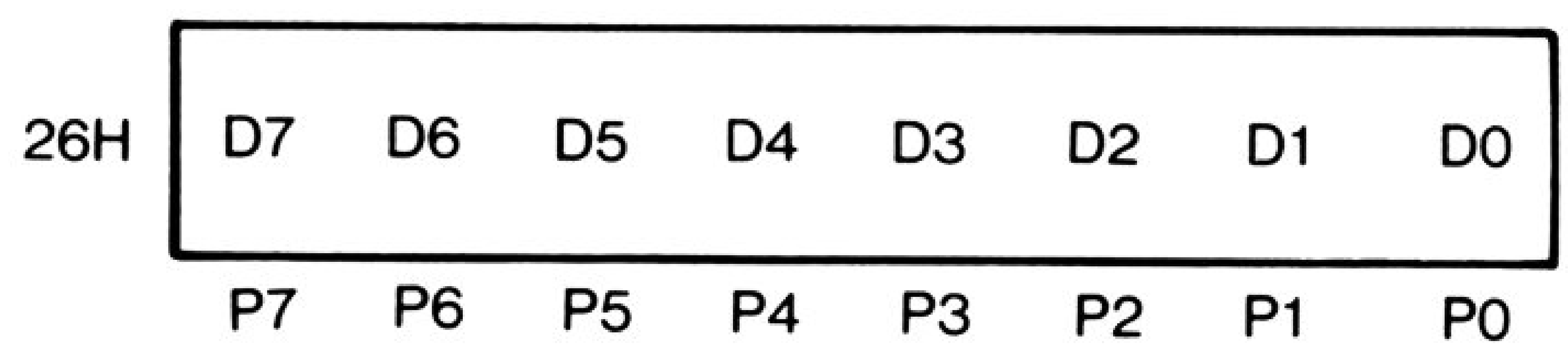
f FM：プリスケーラの項(a－4)参照

例

T 0VA(MAX)＝20.48ms(@ f FM＝600kHz)

T 0VA(MIN)＝0.02ms(@ f FM＝600kHz)

(Ⅱ) TIMER-B：ADDRESS[26H]



タイマBは8ビットのプリセッタブルタイマであり，タイマと同様にカウンタのオーバーフロー時にフラグを立てます．

T　0VB(ms)＝192＊(256－NB)/f FM(kHz)

NB：P7＊2^7＋P6＊2^6＋…＋P1＊2^1＋P0

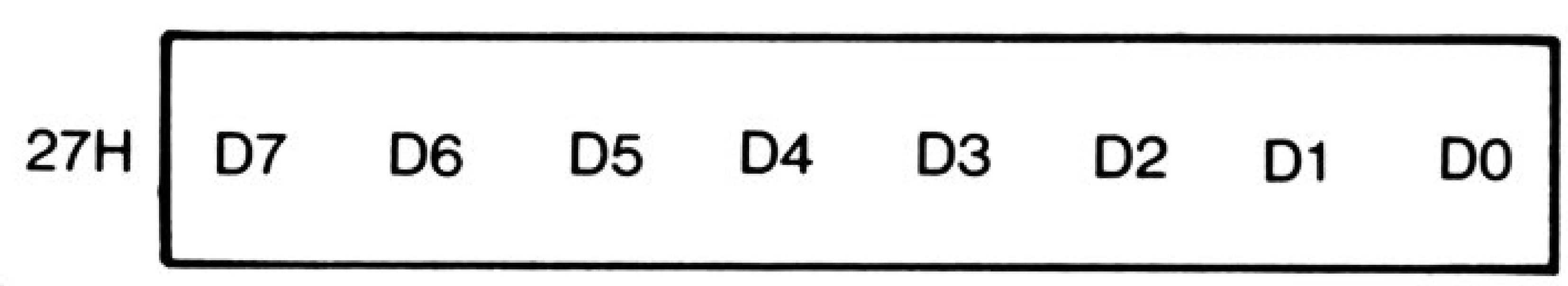
f FM：プリスケーラの項(a－4)参照

例

T 0VB(MAX)＝81.92ms(@ f FM＝600kHz)

T 0VB(MIN)＝0.32ms(@ f FM＝600kHz)

(Ⅲ) タイマコントールおよびチャネル3のMODE設定：ADDRESS[27H]



アドレス27Hでは，タイマA，タイマBの始動・停止・フラグの制御をD0～D5のビットで行います．また，D6, D7の2ビットでチャネル3のMODEを決定します．

各ビットの意味

D0：タイマAの始動・停止の制御をします．このビットに"1"が立ったときに，その立ち上がりで，タイマAのプリセット値をロードしてタイマAを動かします，そして，このビットが"0"になるとタイマAは停止します．

D1：タイマBの始動・停止の制御ビットで，働きはタイマBに対してD0と同様なことを行います．

D2：このビット"1"が立つと，タイマAのオーバーフローを受けてAフラグに"1"を立てます．"0"のときはオーバーフローは受け付けません．

D3：このビットは，タイマBに対してD2と同一の働きをします．

D4：このビットに"1"が立つとAフラグをリセットします．ただし，このビットの"1"は保持されず，Aフラグをリセットすると，クリアされます．

D5：Bフラグをリセットします．リセットデータは保持されません．

D6, D7：この2ビットは上記のD0～D5とは違った意味を持っており，チャネル3のモードを設定します(表3)．

表3　MODE設定

MODE	D7	D6	内　　容
通常	0	0	チャネル3は通常の発音モードです．F-NumberはA2H，A6Hで指定されます．
効果音	0	1	チャネル3のF-Numberは各スロットごとに指定できます．1スロットはA9H，ADH，2スロットはAAH，AEH，3スロットはA8H，ACHのデータを使います．
音声合成	1	0	チャネル3のF-Numberは10の場合と同様ですが，MODEはCSMの音声合成モードとなり，チャネル3のKey-on/offはタイマAでコントロールされます．

(F-Numberについては(b－1)参照)

(a－3)　Key-on/off：ADDRESS[28H]

このレジスタによって，FM12スロットのKey-on/offがコントロールされます．レジスタの下位2ビットがチャネルを指定します．そして，上位4ビットが各チャネルのスロットを指定できます．指定したチャネルスロットに"1"を立てるとKey-onとなり，"0"を書き込むとKey-offとなります．

28H	D7	D6	D5	D4		D1	D0

チャネル割り当て

D1	D0	チャネル
0	0	チャネル1
0	1	チャネル2
1	0	チャネル3

スロット割り当て

D4	第1スロットのON/OFF
D5	第2スロットのON/OFF
D6	第3スロットのON/OFF
D7	第4スロットのON/OFF

例

	D7	D6	D5	D4	D3	D2	D1	D0
チャネル1の全スロットをON	1	1	1	1	*	*	0	0
チャネル2の全スロットをOFF	0	0	0	0	*	*	0	1
チャネル3の1,2スロットのみON	0	0	1	1	*	*	1	0

*はDON'T CARE

(a—4)　プリスケーラ機能：ADDRESS [2DH]・[2EH]・[2FH]

ここでは，FM・SSG各音源の周波数をコントロールします．この3つのアドレスに関してはデータビットはなく，アドレスを指定するだけです．分周数とアドレスの関係は表4のようになります．ただし，FM音源の出力はこの周波数の倍で出されます．

このプリスケーラの注意点として，初期設定時は，FM音源に対しては1/6・SSGに対しては1/4の分周数に設定されています．

表4　入力クロックと内部クロックの関係

2D	2E	2F	FM音源の分周数	SSG音源の分周数	OPNに入力できる最大周波数
—	—	A	1/2	1/1	1.4 MHz
A	—	—	1/6	1/4	4.2 MHz
A	A	—	1/3	1/2	2.1 MHz

(Aはそのアドレスを入力し，—は入力しないことを意味します)

(a—5)　Detune / Multiple : ADDRESS [30H～3EH]

MultipleはF-NumberとDetuneで得られる位相情報に対して，表5で表されるような倍率の情報であり，DetuneはF-Numberにわずかの周波数ズレを与える情報です．

		MSB		LSB	MSB			LSB
30H～3EH		D6	D5	D4	D3	D2	D1	D0
		Detune			Multiple			

表5　倍率

Multiple情報	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
倍　　率	1/2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Detune/MultipleよりSSG-EGまでの7種類のFM制御情報は，各スロットごとに決めることができます．各アドレスとスロットチャネルの対応は，表6−aのとおりです．表からもわかるように，チャネルとアドレスの下位2ビットが対応しているため，アドレスの下位2ビットが"11"になることはありません．また，2スロットと3スロットの順序が逆になっています．

表6−a　アドレスとチャネルスロットの関係

ADDRESS	チャネル	スロット	ADDRESS	チャネル	スロット
＊0H	1	1	＊8H	1	2
＊1H	2	1	＊9H	2	2
＊2H	3	1	＊AH	3	2
＊4H	1	3	＊CH	1	4
＊5H	2	3	＊DH	2	4
＊6H	3	3	＊EH	3	4

（＊は3より9までの値）

また，F-Number/BlockとSelf-Feedback/Connectionのデータはチャネルごとに決められるものであり，そのデータは，対応する4スロットに共通です．アドレスとチャネルの関係は次のとおりです．

表6−b　アドレスとチャネルの関係

ADDRESS	チャネル	ADDRESS	チャネル	ADDRESS	チャネル
＊0H，＊4H	1	＊1H，＊5H	2	＊2H，＊6H	3

（＊はA あるいはB）

(a−6)　Total　Level：ADDRESS［40H～4EH］

40H～41H	／	D6	D5	D4	D3	D2	D1	D0

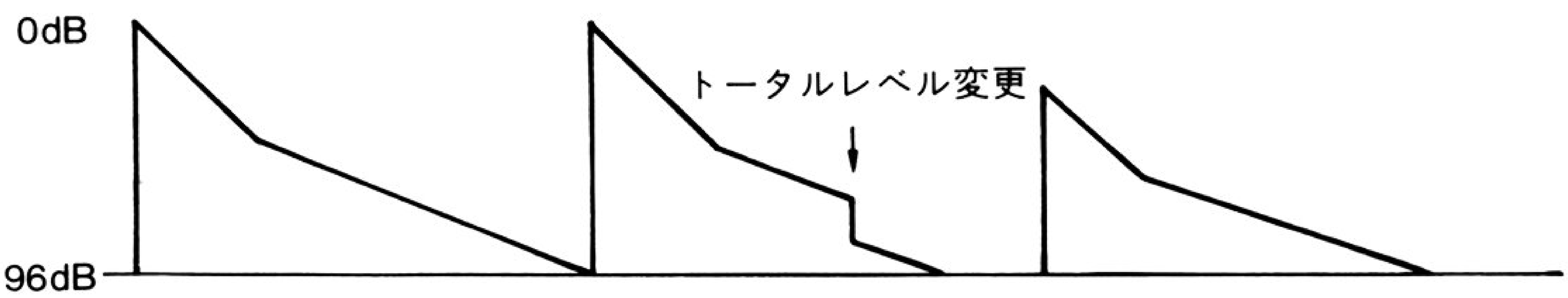
トータルレベルとは，エンベロープジェネレータの出力に対して，減衰量を加算し，変調度(音色)および音量の制御をするために用いられます．減衰量は最小分解能を0.75dBとして，各ビットの重み付けは表7のとおりです．

表7　トータルレベルでの減衰量

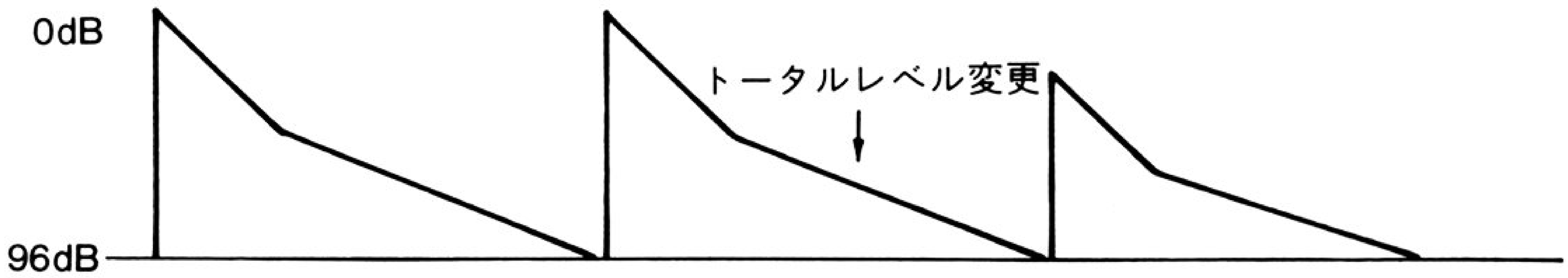
データビット	D6	D5	D4	D3	D2	D1	D0
減衰量(dB)	48	24	12	6	3	1.5	0.75

トータルレベルには，もう一の働きがあります．それは，CSMのモードを選択した場合です．このときトータルレベルは3チャネルの音(CSM合成音)に対し，エンベロープのイニシャルレベルとなります．したがって，エンベロープの途中でトータルレベルを変えても，エンベロープの減衰量は変化せず，次のKey-on時のレベルが変わります．

• 通常のエンベロープ



• CSMモードのエンベロープ



(a−7)　Key Scale/Attack Rate：ADDRESS［50H～5EH］

・Key Scaleは，エンベロープのRateを音程によって変化させるために設けられています．Key Scaling後のRateは，次式によって表されます．式中のRは入力したRateであり，RksはKey Scaling量でこの値は表8のとおりです．

Rate＝2＊R＋Rks

(ただしR＝0の場合は，Rate＝0とする)．

表8　RateのKey Scale

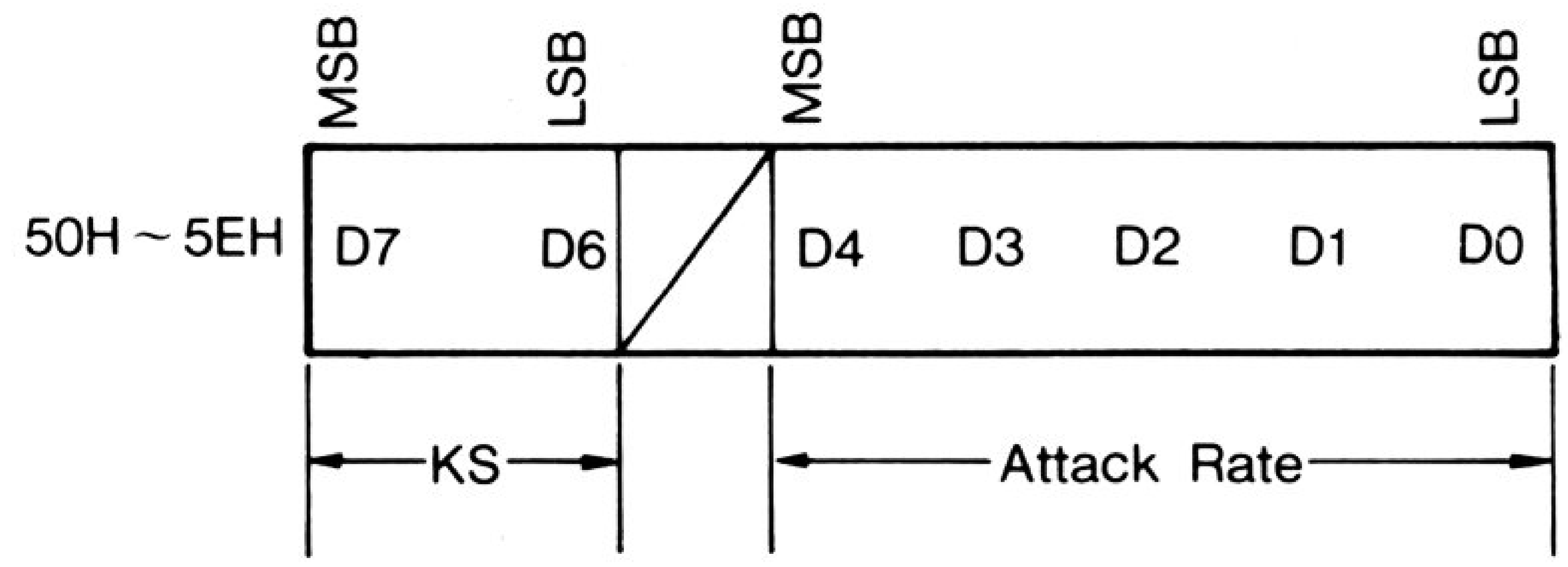
KS ＼ Key-Code	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
2	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7
3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

KS ＼ Key-Code	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3
1	4	4	4	4	5	5	5	5	6	6	6	6	7	7	7	7
2	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15
3	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Key-Codeについては(b−1)を参照

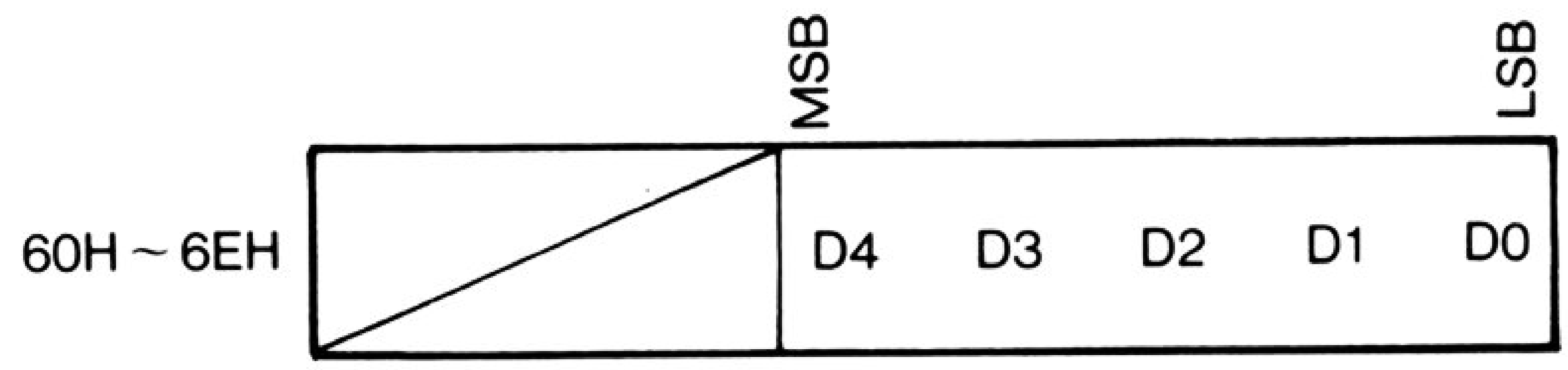
Attack Rateは，音の立ち上がりの時間を設定します．

なお，エンベロープの形状は(e)のエンベロープジェネレータの項を参照してください．



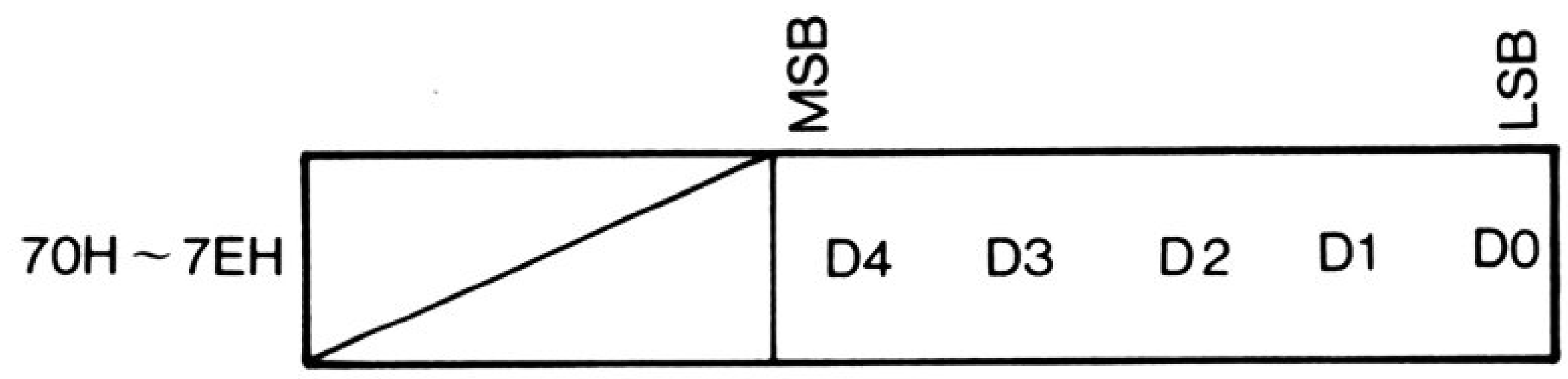
(a−8)　Decay　Rate：ADDRESS[60H～6EH]

Decay時のRateを設定します．



(a−9)　Sustain　Rate：ADDRESS[70H～7EH]

Sustain時のRateを与えます．



(a−10)　Sustain　Level/Release　Rate：ADDRESS[80H～8EH]

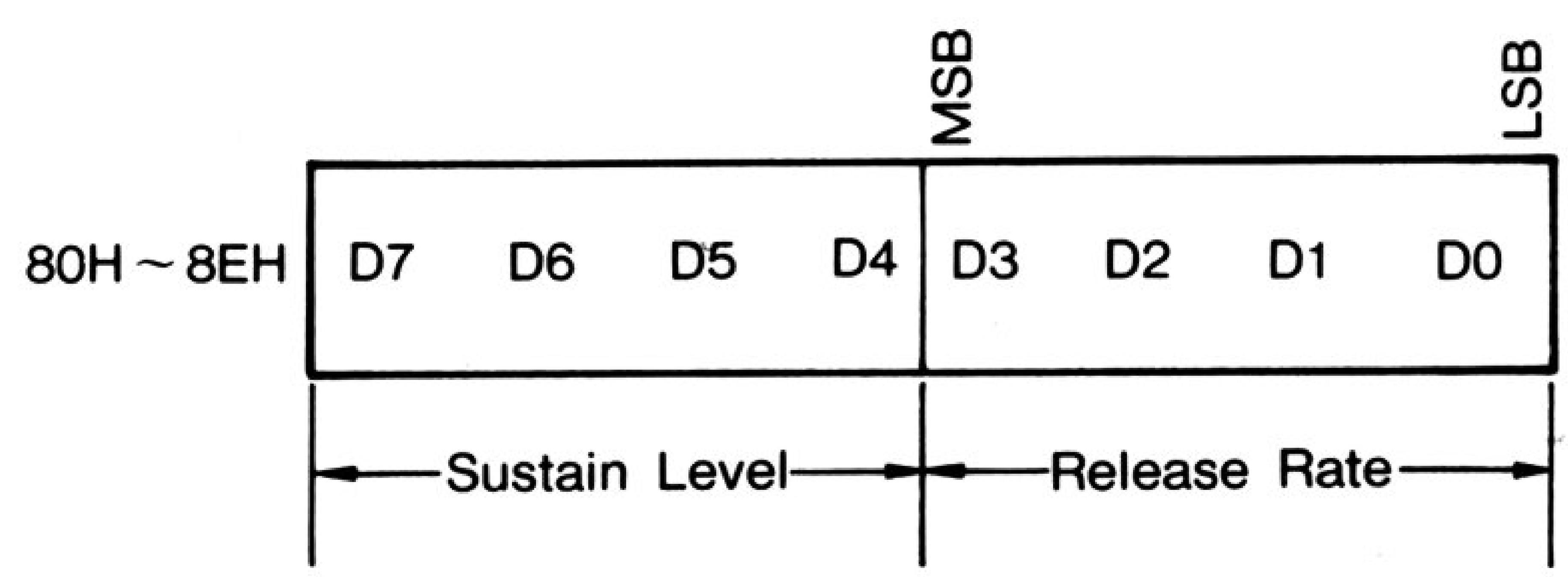
Sustain LevelはDecay Rateと，Sustain Rateの切り替え点を与えます．その重み付けは表9で表され，値は減衰量を示します．

表9　サスティンレベルと重み付け

データビット	D7	D6	D5	D4
減衰量(dB)	24	12	6	3

(ただし，D4～D7がすべて"1"の場合は，93dBとなります)．

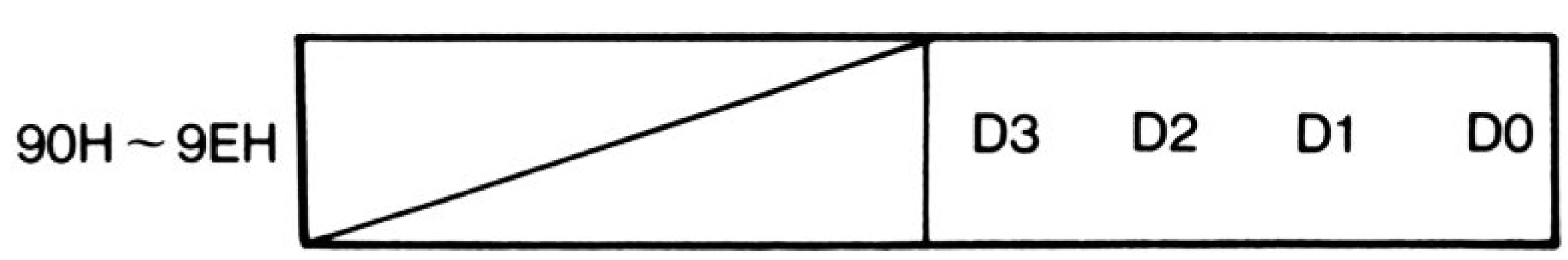
Release RateはKey-offのときのRateを与えます．ただし，Release RateはLSBが"1"に固定されているため，他のRateと異なって4bitsしか入力できません．



(a−11)　SSG-Type　Envelop　Control：ADDRESS[90H～9EH]

SSG-Type Envelopとは，後述のSSG音源で作られるエンベロープと同一形状を与える機能です．この4ビットのデータ中，D3がこのタイプのエンベロープをとるか否かのスイッチとなっており，残り3ビットがデコードされ，図1の形状のエンベロープを発生します．

このエンベロープが使われた場合の注意点は，Rateの使い方が通常の場合とは異



＊　Attack Rateは1FHに固定されていなければなりません．

なっていることです．このモードでは，パラメータはDecay, Sustain, Releaseの各Rateです．この中でRelease Rateは，Key-off時のみ有効であるのは，通常モードと変わりありません．したがって，Key-on中に音量が増加したり，減少したりする場合の制御は，Decay およびSustainの2つのRateでなされます(音量の増加は減少の場合を逆にたどっていく形になります)．もう一つ通常モードと異なる点は，減衰時間の違いです．通常モードより変化が速くなります．

このエンベロープはエンベロープジェネレータ出力の波形であり，対数出力です．OPNの出力では，リニアに変換されますので増減の傾きは変化します．

SSG-Type EGを使ったときの注意点としては，次の2つがあげられます．

(1) Attack Rateは，通常1FHにしなければなりませんが，08H, 09H, 0BHなどは普通のエンベロープ時のRATEで変化します．その他のものについては十分な制御はできません．

(2) Decay Rate, Sustain Rate, Sustain Levelはエンベロープが UP時，またはDOWN時では異なります．

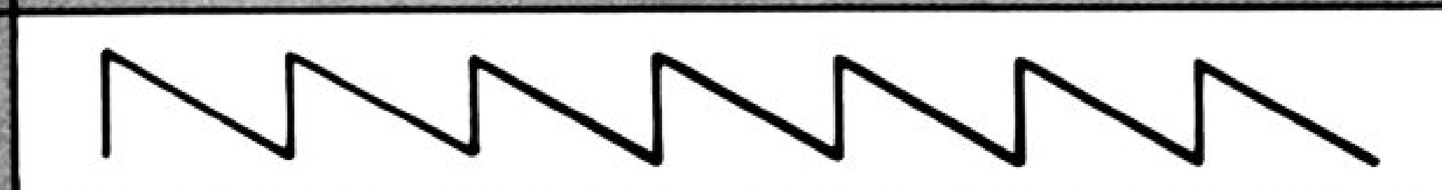
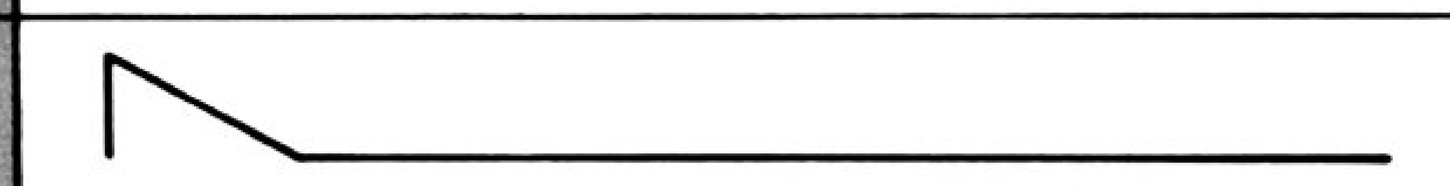
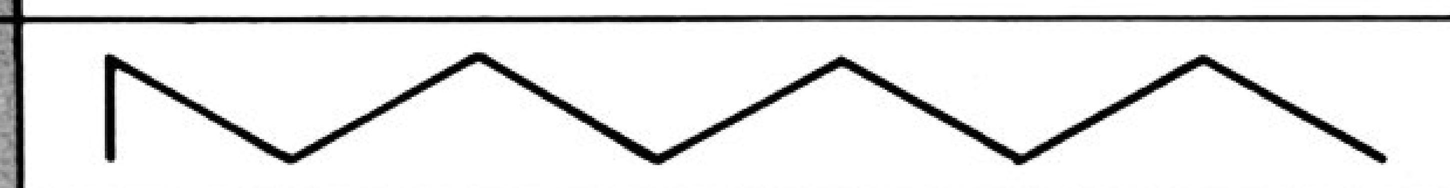
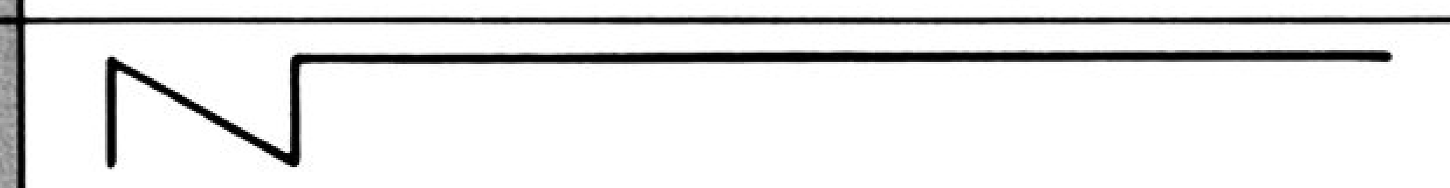
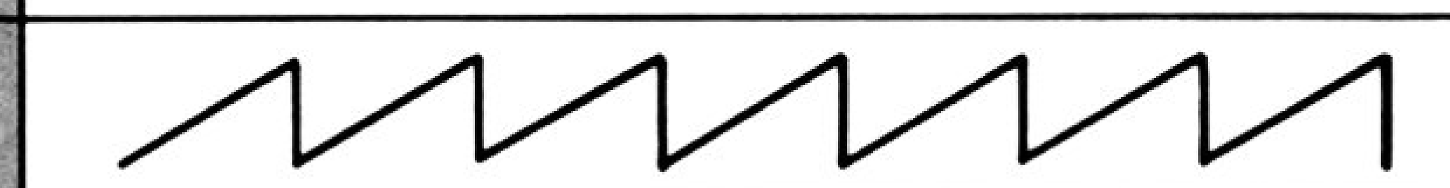
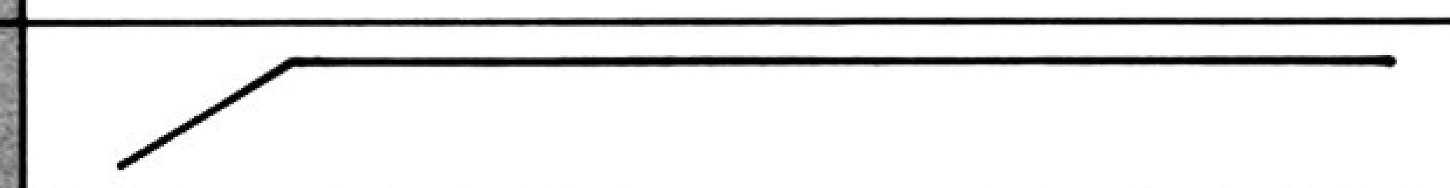
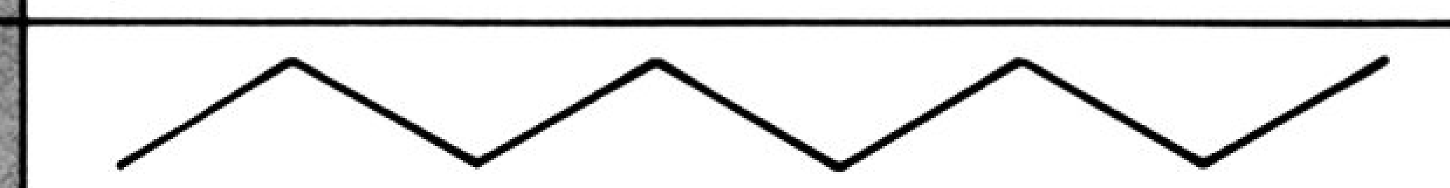
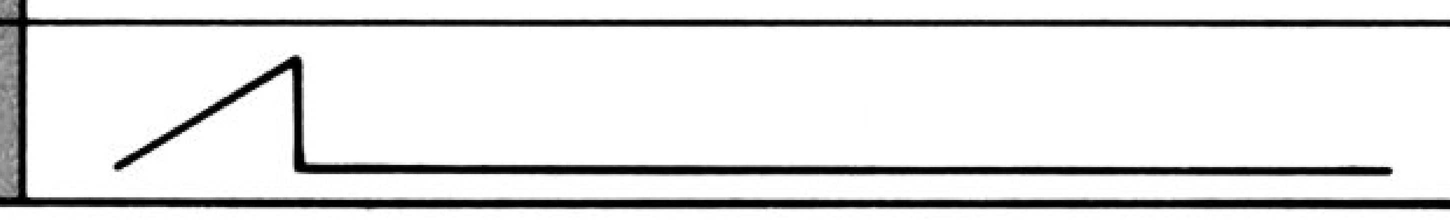
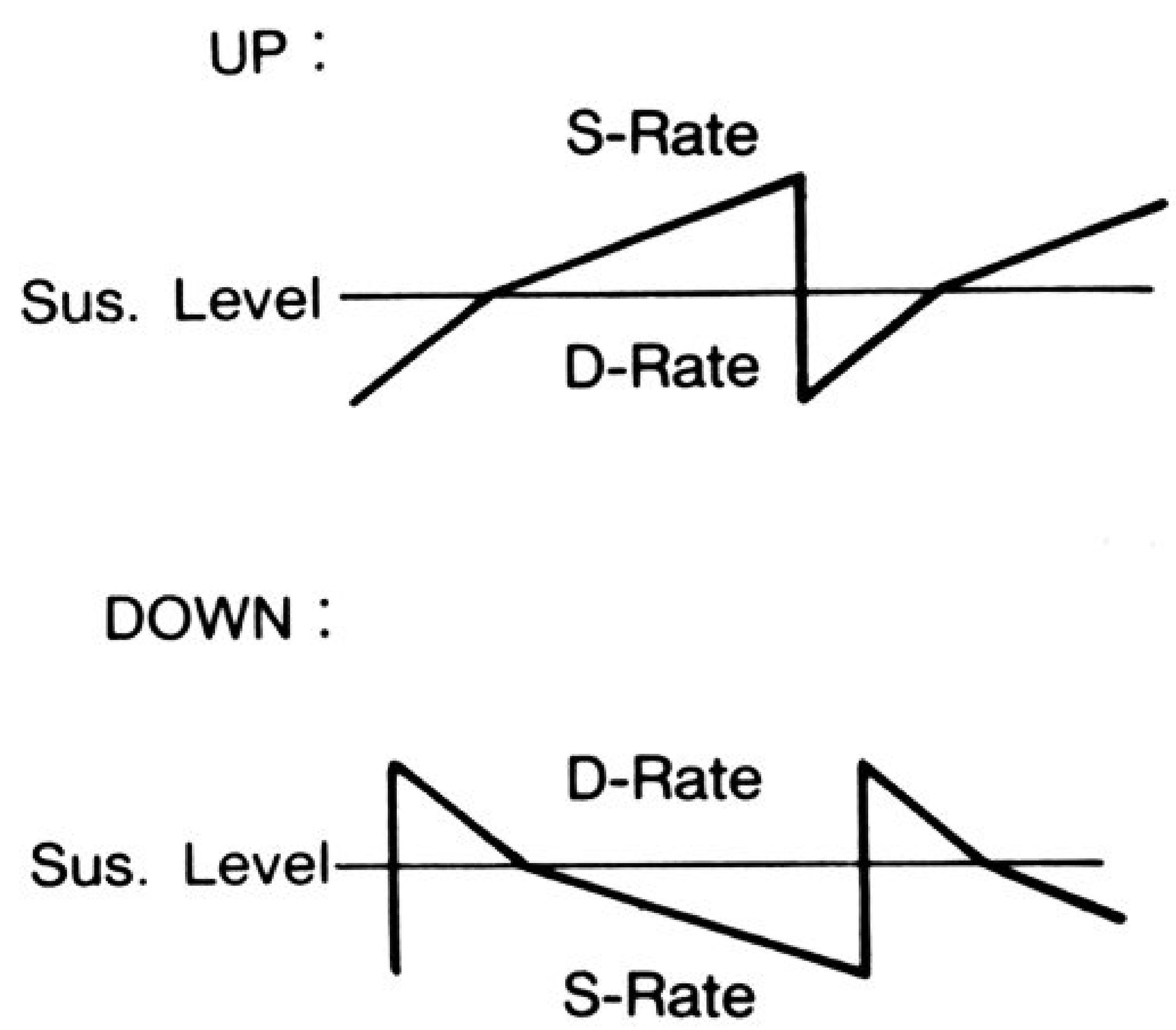
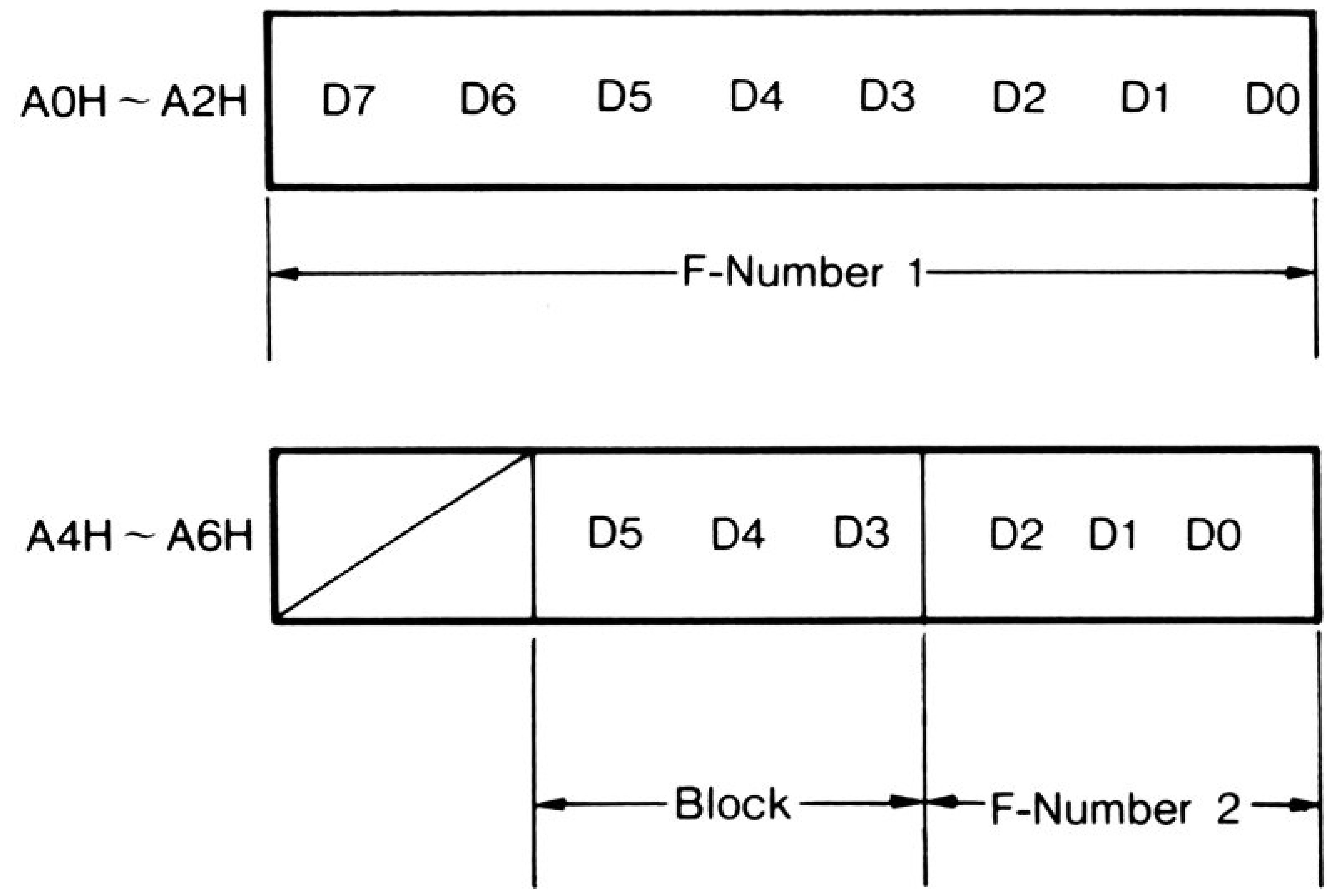
D3 D2 D1 D0	エンベロープ波形
1 0 0 0	
1 0 0 1	
1 0 1 0	
1 0 1 1	
1 1 0 0	
1 1 0 1	
1 1 1 0	
1 1 1 1	

図1　SSG-Typeのエンベロープ

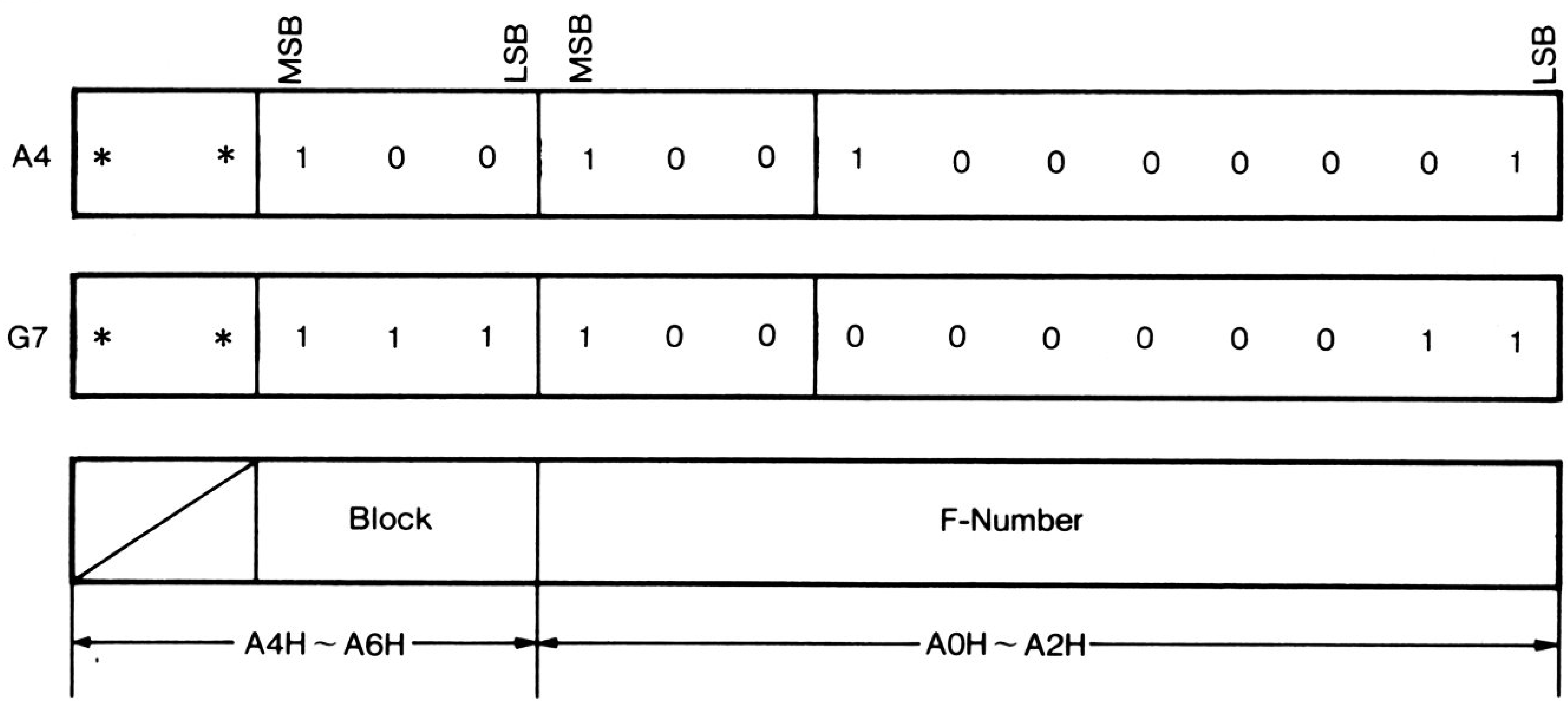


(a—12) F-Number と Block データ：ADDRESS[A0H～A2H, A4H～A6H]

このアドレスで，F-Number と Block のデータがセットされます．データをセットする場合の注意点としては，まず最初に Block と F-Number の上位 3 ビット（F-Number2)をセットし，次に F-Number の下位 8 ビット(F-Number1)を入力します．F-Number と Block の内部のレジスタへのローディングは，F-Number1 がセットされたときのみ動作しますから，データの入力の順序が異なりますと，Block と F-Number2 のデータは違ったデータとなります．



例 音程データ(@φM＝1.2MHz)



(a—13) 3 チャネルの F-Number と Block：ADDRESS[A8H～AAH, ACH～AEH]

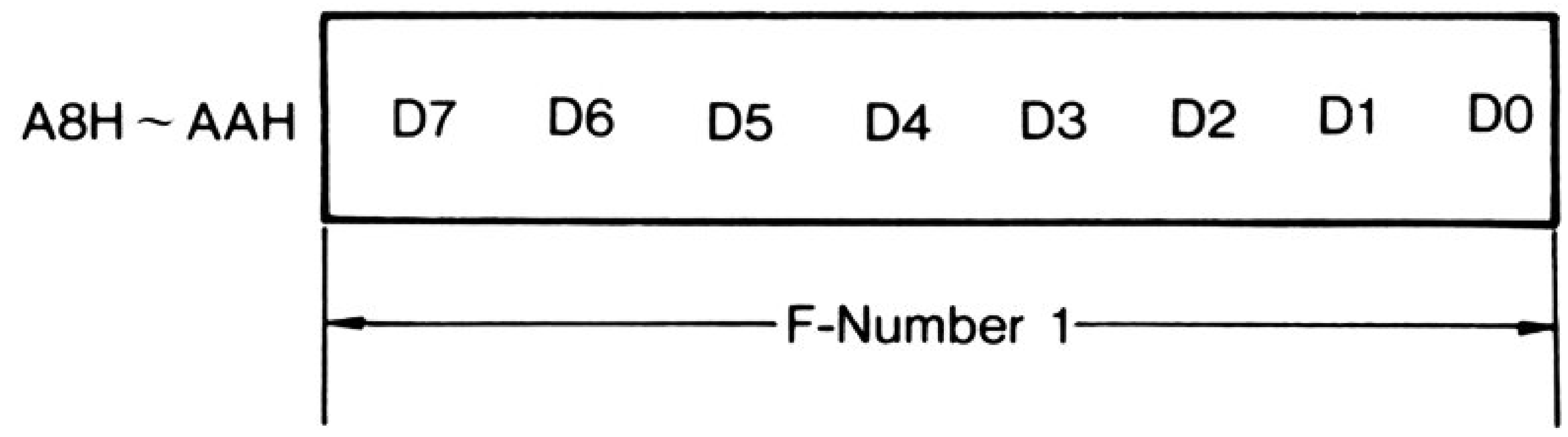
このアドレスでセットされる F-Number と Block は，効果音モードや音声合成モードで使われるデータです．データセットの方法は(a—12)で述べた方法と同様で，先に Block, F-Number2 のデータを，次に F-Number1 のデータをセットします．

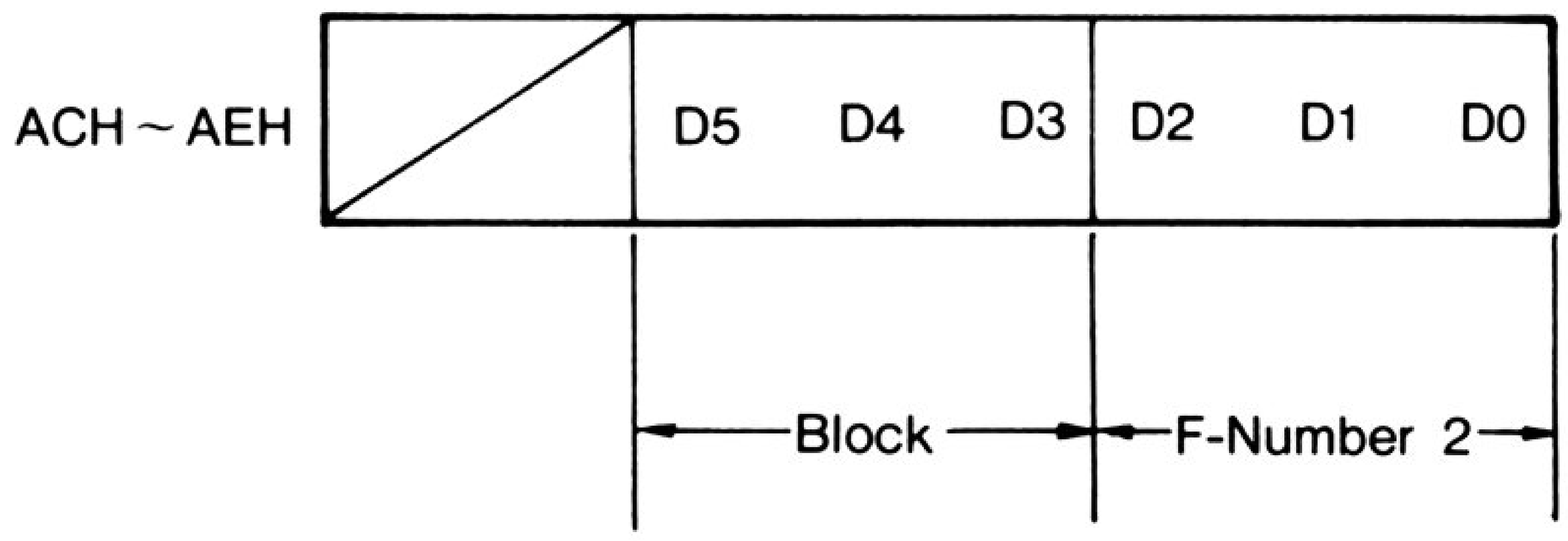
ここでセットしたデータとスロットの関係は，次のようになります．

右のスロットとアドレスとの関係は，効果音モードと音声合成モードのときに適用

スロット番号	F-Number/Block のアドレス
1	A9H, ADH
2	AAH, AEH
3	A8H, ACH
4	A2H, A6H

されるのであって，通常モードでは全スロットとともA2HとA6Hのアドレスで与えられます．





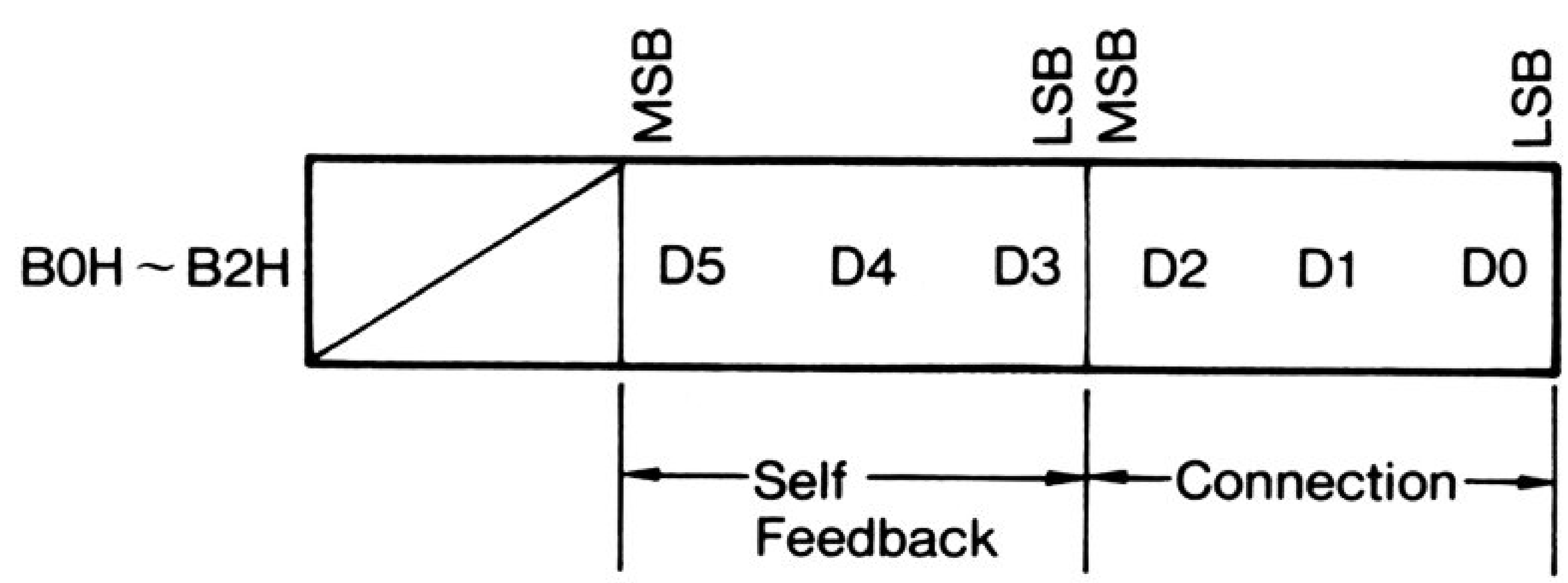
(a—14) Self-Feedback / Connection : ADDRESS[B0H～B2H]

Self-Feedbackというのは，各チャネルの第1スロットの変調度を決めるデータです．つまり，第1スロットは自分自身の出力を変調信号としているため，その変調度合をこのデータで制御します．変調度は，表10に示すとおりです．

Connectionは4つのスロットをどのように変調波と被変調波に分け，どういうふうに組み合わせるかを指定するデータです．それらの組み合わせは全部で8つあり，その構成については，BASICリファレンスマニュアル　4章　CMD VOICEのアルゴリズム表を参照してください．

表10　変調度

D3～D5	0	1	2	3	4	5	6	7
変調度	OFF	$\pi/16$	$\pi/8$	$\pi/4$	$\pi/2$	π	2π	4π



(b) **フェイズジェネレータ(PG)**

フェイズジェネレータは，必要な周波数に応じた増分を単位時間(DACサイクル)ごとにアキュムレートして位相値を得る回路です．この増分は，レジスタから送られてくる周波数情報(BLOCK/F-Number/Detune/Multiple)から作られます．4つある周波数情報の中で柱となるのは，F-Numberであり，これにより1オクターブ中の音程が決定されます．そこで，まずF-Numberから説明します．

(b−1) **F-Number/BLOCK**

位相の増分は発音したい周波数と，入力クロック周波数(サンプリング時間)が決まると，すぐに求めることができます．これは次式になります．

$\triangle P = f\ Mus * 2^{20} / f\ SAM$

$f\ SAM = fM/(12 * n)$　n=2, 3, 6

f Mus： 発音したい周波数

f SAM：サンプリング周波数

f M： 入力クロック周波数

n： プリスケーラコントロールで得られる定数〔(a−4)参照〕

これで位相の増分は求まりますが，この増分を管理するのはビット数が多くて大変です．そこで，増分は1オクターブ分のデータのみを管理できるビット数に制限し，指定したオクターブごとにその増分をシフトすることにより，位相を求めることとします．これにより，オクターブ情報の指定をBlockデータに，増分をF-Numberに入力すれば全音域をカバーできることになります．以上の操作とF-Numberを11ビットで表現することにしたため，結局F-Numberは次式で求められます．

$F = (f\ Mus * 2^{20} / f\ SAM) / 2^{b-1}$

(bはオクターブデータ)

例

3.6MHzの入力クロックでA4(440Hz)のF-Numberを求める．

$F\ A4 = (440 * 2 / 50000) / 2^{4-1} = 1153$

(誤差−0.65セント)

次に，Key Scaleを行うときに問題とな

る1オクターブ内の分割について少し説明します．

Key Scaleは，通常1オクターブ内を4分割（3音ずつの組）して行うことが多くありますが，F-Numberを用いると一定のセントで分割することは困難となります．そこで，OPNではF-Numberの上位4ビットをデータとして分割を行っています．ただし，この方法では入力周波数によって分割が異なります．

1オクターブ内の周波数分割

$N4 = F11$

$N3 = F11 \cdot (F10 + F9 + F8) + \overline{F11} \cdot F10 \cdot F9 \cdot F8$

以上の操作で得たN4，N3の2ビットとオクターブデータの3ビットの計5ビットでKey Scaleを行います．

	Block			Note	
	B3	B2	B1	N4	N3
Block，Noteを分離した重み付け	D2	D1	D0	D1	D0
Block，Noteをまとめた場合の重み付け	D4	D3	D2	D1	D0

(b—2) Detune / Multiple

レジスタの項でも説明したように，Detuneは与えられたF-Numberに対して周波数変位を与えるために，周波数に応じた定数を加算あるいは減算します．またMultipleは，Detuneの操作を受けたF-Numberに指定された倍率を乗ずる働きをします．

(b—3) アキュムレータ

DetuneおよびMultipleの回路を通ったF-Numberを累積加算し，その結果をオペレータに送ります．このアキュムレータは加算のみを行い，オーバーフローが生じていても無視します．これは，オーバーフローが起こったことは弧度法でいう2πを通過したにすぎないからです．

付録3.2 SSG音源部

SSG音源部は，DACを内蔵した音源であり，そのレジスタアレーにデータを書き込むだけで楽音を発生します．SSG内部のレジスタアレーはFM音源部と異なり，READ/WRITEが可能です．以下，各レジスタごとにその機能を説明していきます．

(a) トーンジェネレータコントロール：ADDRESS[00H～05H]

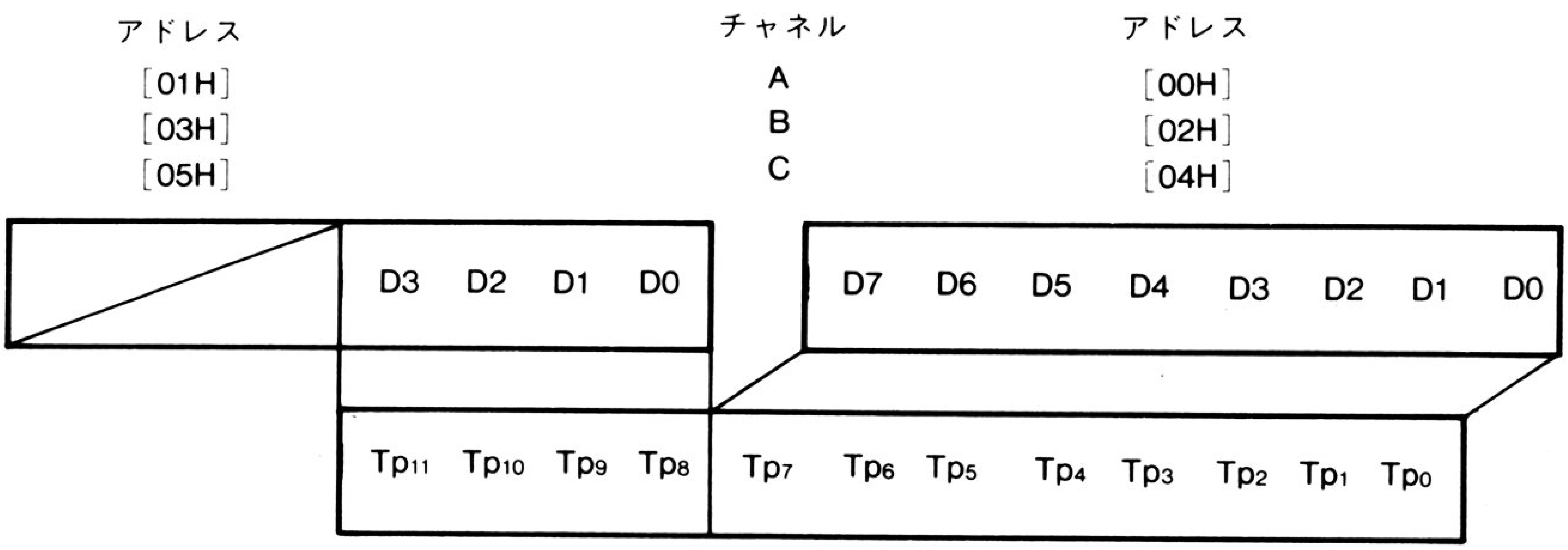
SSG部もFM音源部と同様に，3チャネルの楽音を発生することができます．この3チャネルの楽音の周波数を設定するのがこのレジスタです．楽音周波数は12ビットで構成されるため，1チャネルにつき2つのアドレスを必要とし，そのビットの対応は下記のとおりです．また，発信周波数は次式で求められます．なお，出力波形は1:1の短形波です．

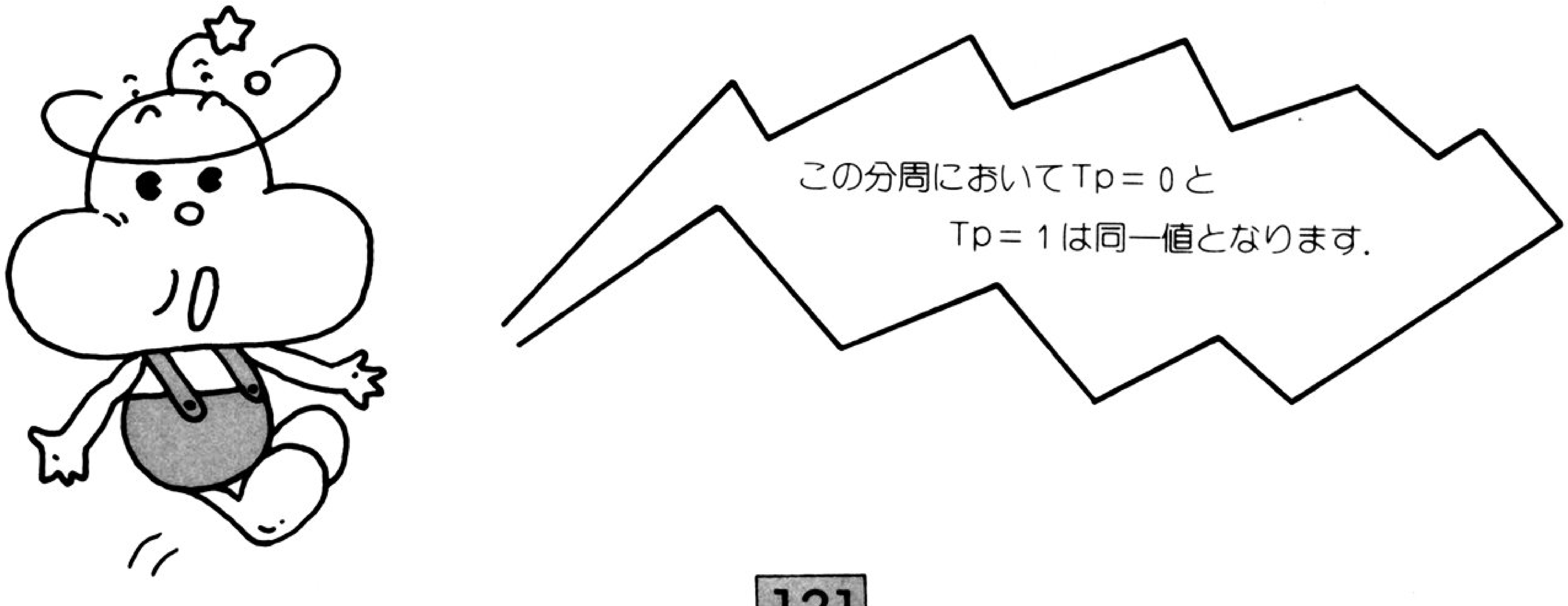
$f\ tone = f\ m/(8*n*Tp)$　$n=1, 2, 4$

fm: 入力クロック周波数

n: プリスケーラコントロールで定められる定数

Tp: 発信周波数設定値





(b) ノイズジェネレータコントロール：ADDRESS[06H]

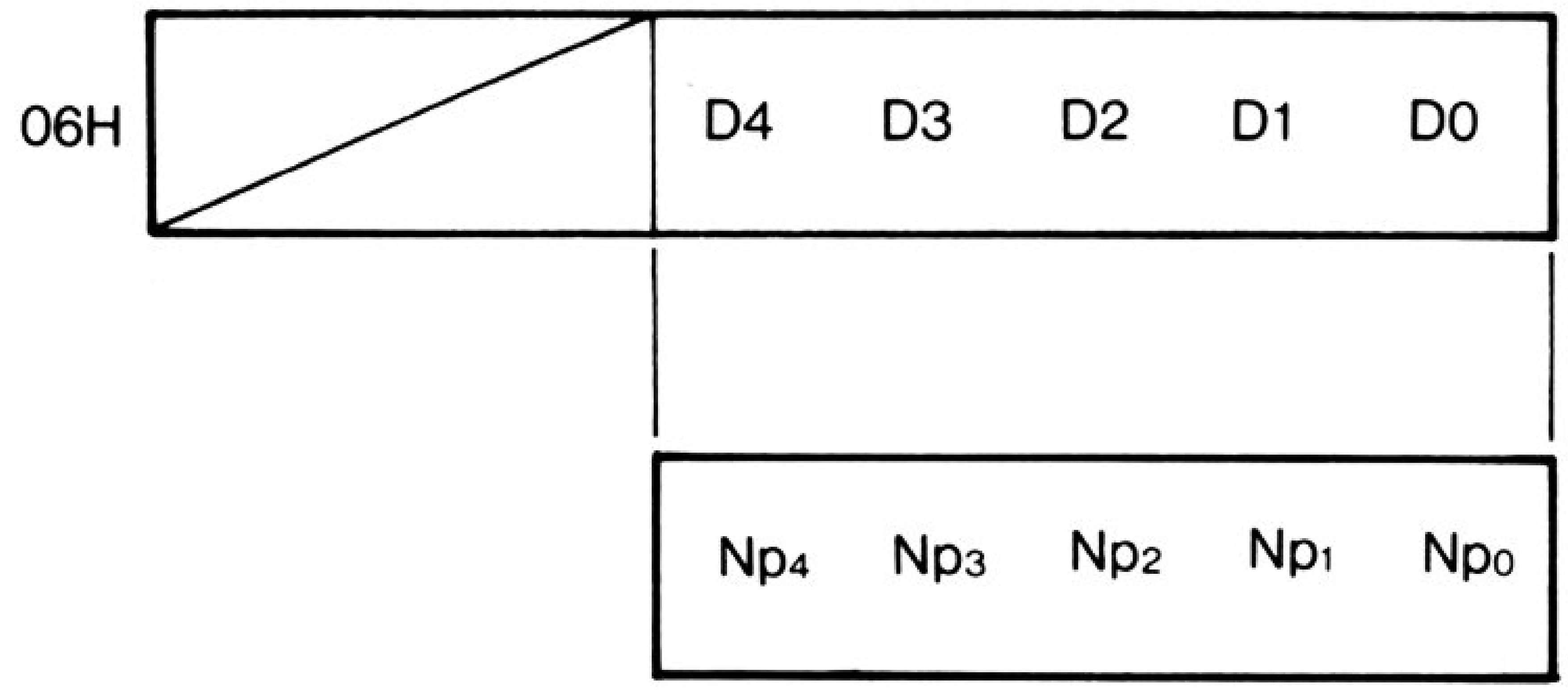
ノイズ源は系列長の疑似ランダムノイズです．そして，この系列の発生をコントロールするのがこのレジスタの働きです．系列発生周期(ノイズ周波数)はレジスタの下位5ビットで決められ，次式で求めることができます．

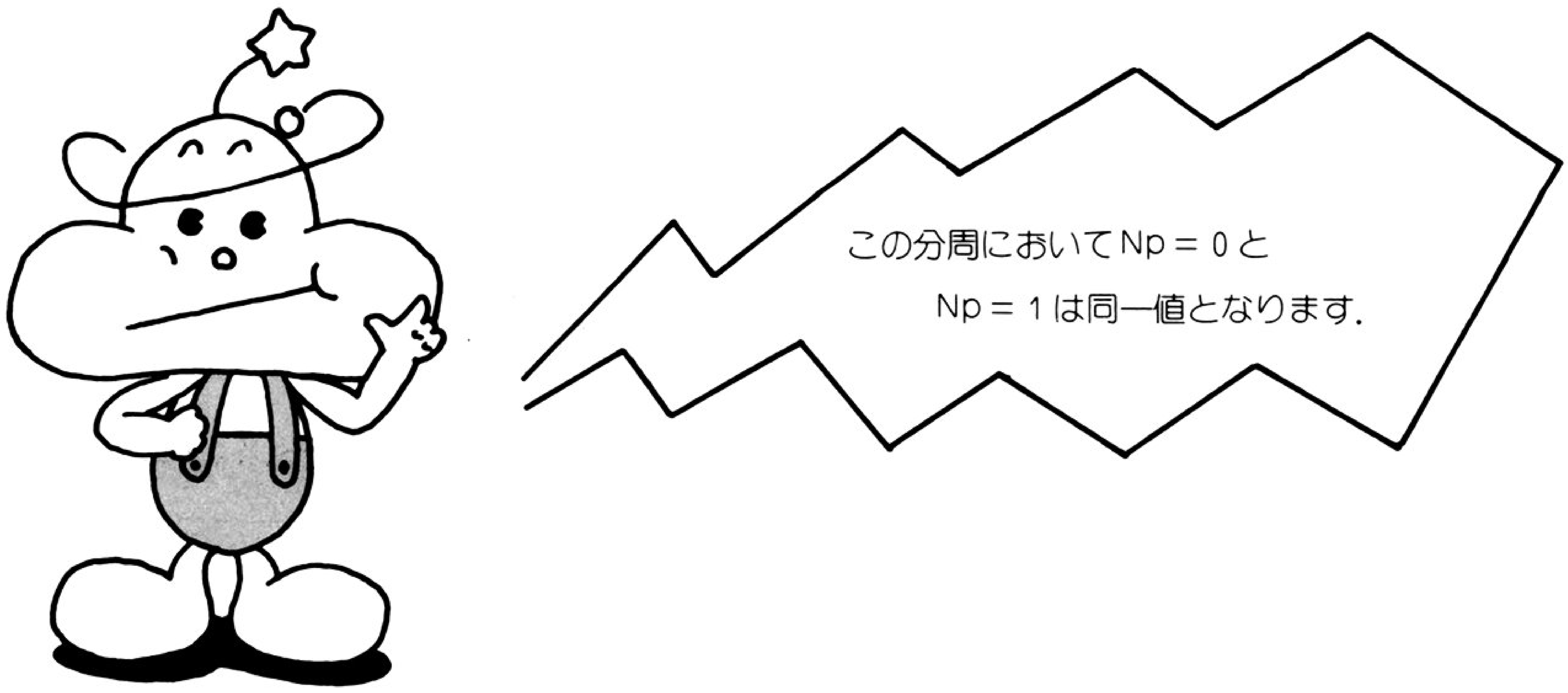
$fN = fM/(8*n*Np)$　$n=1, 2, 4$

fM: 入力クロック周波数

n:　プリスケーラコントロール設置値

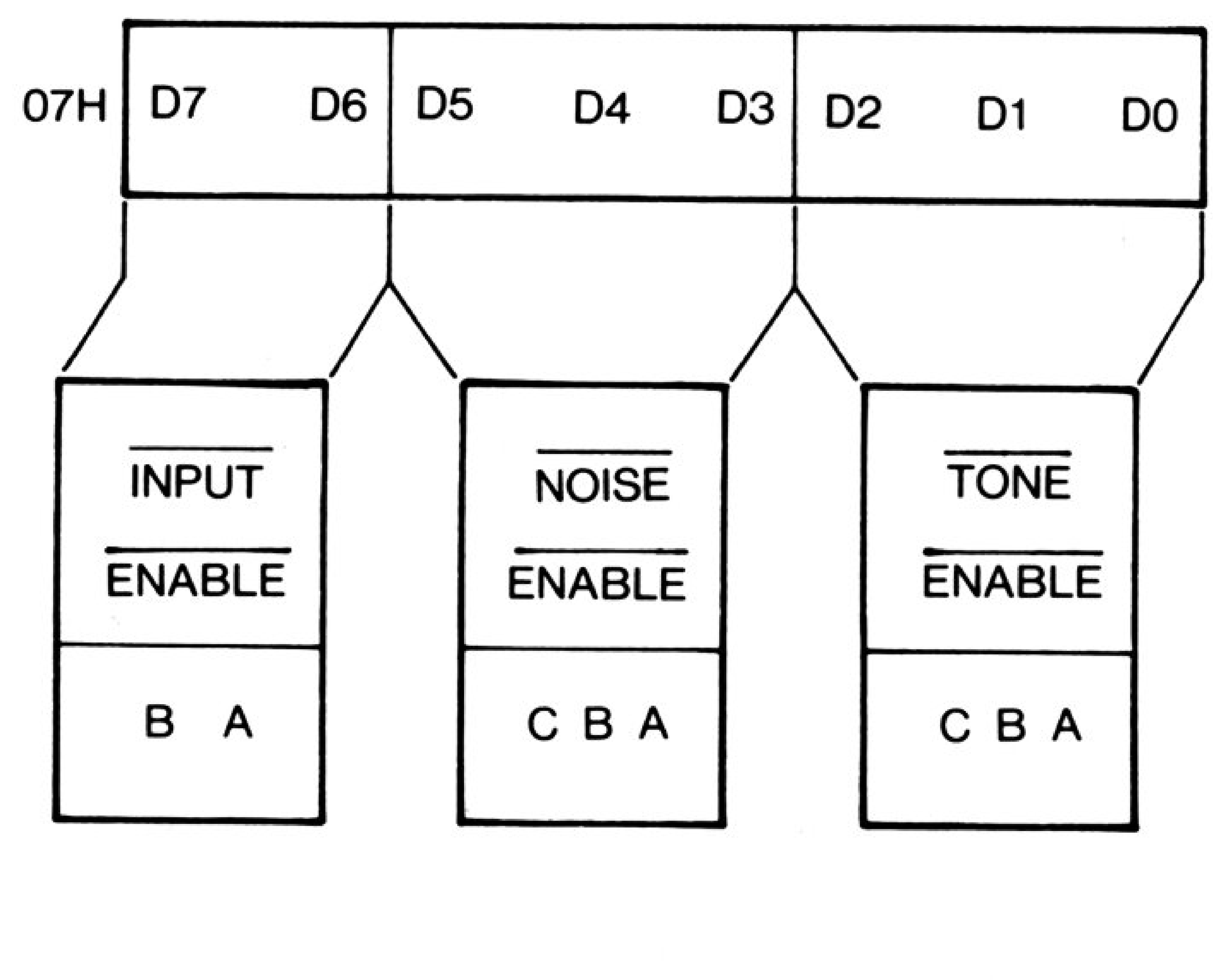
Tp: ノイズ周波数設定値





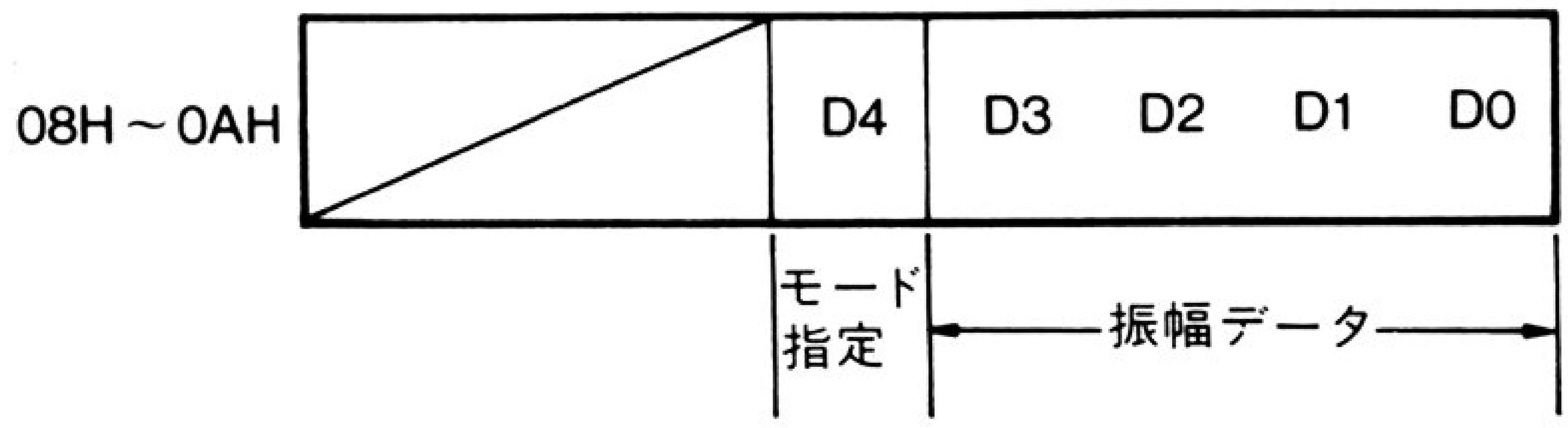
(c) ミキサI/Oコントロール：ADDRESS[07H]

このレジスタは3つのノイズ/トーンミキサと2つの汎用I/Oポートのコントロールをします．ミキサは3つのチャネルごとに，ノイズとトーンともに出力するのか，いずれか一方だけを出力するのか，あるいは両方とも出力しないのかをビット0～5で制御します(データ "0" で発音します)．また，2つのI/Oポートの入出力制御はビット6と7でなされます(データ "0" でINPUTです)．



(d)　振幅コントロール：ADDRESS[08H～0AH]

A, B, C 3つのチャネルに対応するD/Aコンバータの振幅は，このレジスタによって決められます．レジスタのビット4が振幅のモードを決めるデータとなっており，"0"のとき固定振幅モードとなり，残りの4ビットで決められる値をとります．また"1"の場合は可変振幅モードとなり，後述するエンベロープコントロールの出力で決まる振幅となります．なお，D/Aコンバータは，入力5ビットの能力を持っており，固定振幅モード時の4ビットの振幅データは，D/Aコンバータの入力上位4ビットに対応します．



(e)　エンベロープジェネレータコントロール：ADDRESS[0BH～0DH]

変化のあるエンベロープを発生するために，2つのエンベロープコントロールがあります．その1つは，アドレス0BHと0CHで定められるエンベロープ周波数を制御することであり，もう一つは，アドレス0DHのエンベロープの形状を制御することです．

0BH	D7	D6	D5	D4	D3	D2	D1	D0
	F_{T7}	F_{T6}	F_{T5}	F_{T4}	F_{T3}	F_{T2}	F_{T1}	F_{T0}

0CH	D7	D6	D5	D4	D3	D2	D1	D0
	C_{T7}	C_{T6}	C_{T5}	C_{T4}	C_{T3}	C_{T2}	C_{T1}	C_{T0}

MSB															LSB
C_{T7}	C_{T6}	C_{T5}	C_{T4}	C_{T3}	C_{T2}	C_{T1}	C_{T0}	F_{T7}	F_{T6}	F_{T5}	F_{T4}	F_{T3}	F_{T2}	F_{T1}	F_{T0}

まず，エンベロープ周波数について説明します．指定されたアドレスの2バイト16ビットのデータで入力周波数を分周することにより，エンベロープ周波数は求まります．それは次式で与えられます．

$$fE = fM/(128*n*Ep) \quad n=1, 2, 4$$

$$Ep=256*CT+FT$$

fM: 入力クロック周波数

n:　プリスケーラコントロール設定値

CT: 0CHで与えられるエンベロープ周期設定値

FT: 0BHで与えられるエンベロープ周期設定値

このエンベロープを作るカウンタの上位5ビットがD/Aコンバータへ送られ，振幅データとなります．

次にエンベロープの形状について説明します．先に述べたように，エンベロープはエンベロープカウンタの上位5ビットで決まります．したがって，ここでコントロールするのは，この上位5ビットのカウント

0DH		D3	D2	D1	D0
		C	ATT	ALT	HLD

の方法ということになります．アドレス0DHの下位4ビットがこれを決定します．各ビットの機能は次のように定義され，それらの組み合わせにより，図2-aで与えられるエンベロープの形状が得られます．

C：　"1"にセットされたときはHLDのデータで定められる状態になり，"0"の場合は，エンベロープジェネレータは1サイクル経過後すべて"0"にリセットされます．

ATT：　"1"にセットされたときはエンベロープジェネレータはすべて"0"からすべて"1"へカウントアップし，"0"であればすべて"1"からすべて"0"へカウントダウンします．

ALT：　"1"にセットされたときは，各サイクルごとにエンベロープカウントの方向を逆転します．

HOLD："1"のときは，エンベロープカウンタを1サイクルカウントした後，カウンタの最終値をホールドします．ただし，ALTも"1"のときは，直前のサイクルの初期カウント値にリセットされます．

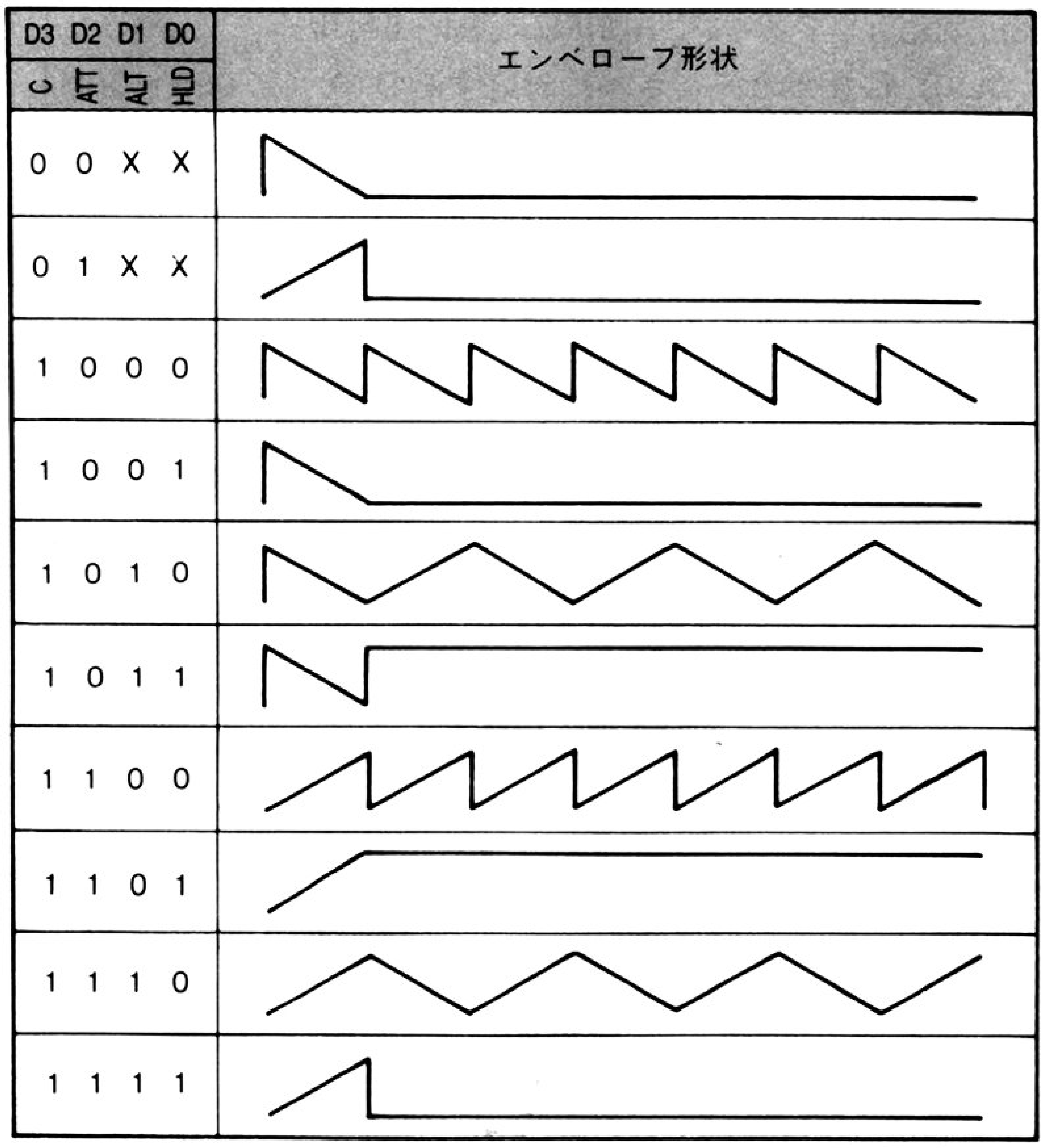


図2-a　SSG部のエンベロープ波形

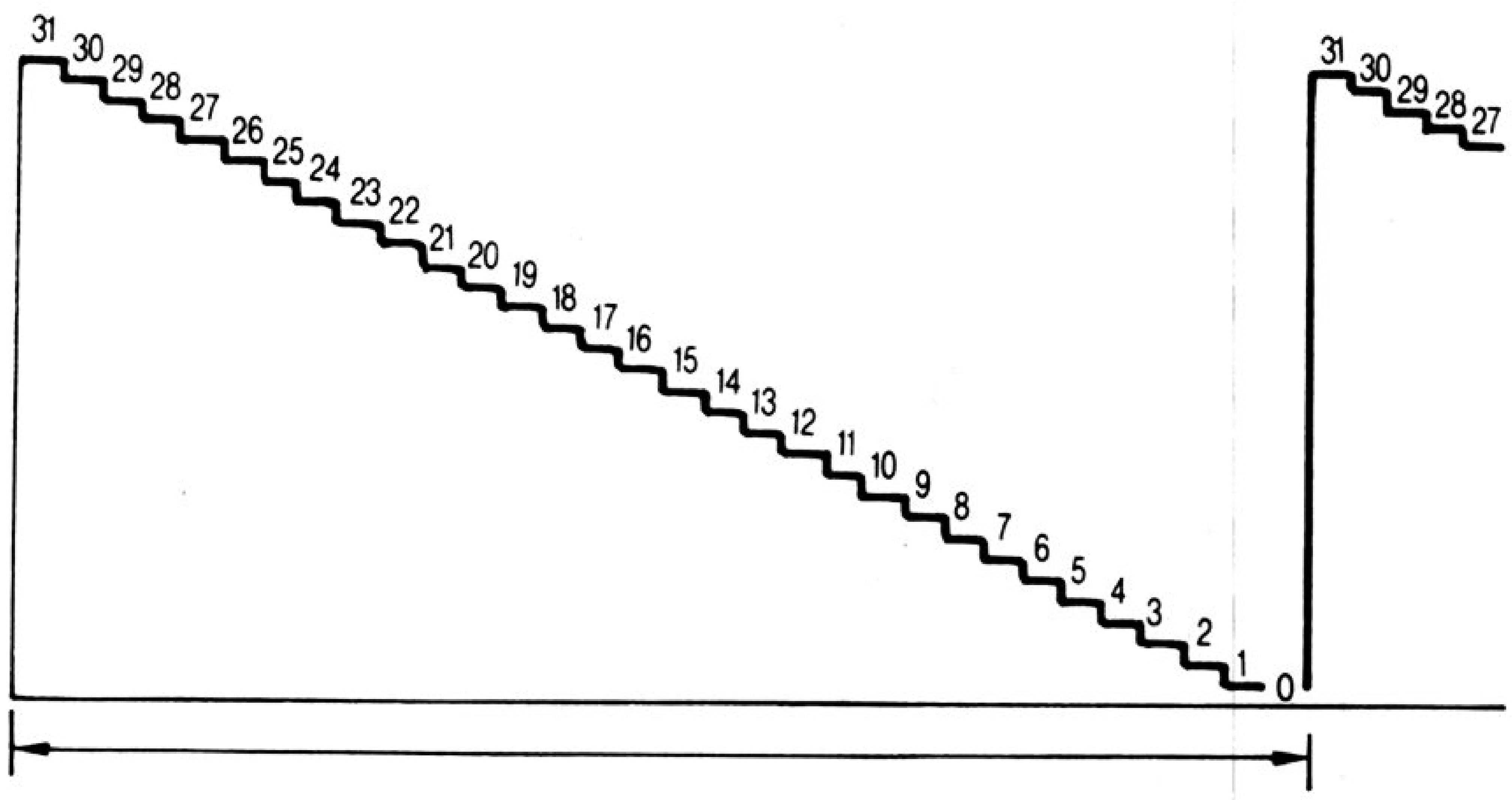


図2-b　1000データの詳細

資　料

資料では，
N88-BASIC，N88-日本語BASICを使ううえでの，
補足的な資料と，プログラミングの際に
頻繁に使われる表を集めました．

資料1．PC-8800シリーズの各機種のBASIC

資料1.1 各BASICの比較

PC-8801mkⅡMRは，従来のPC-8800シリーズの各機種との互換性を保つための2つのBASIC(N88-BASIC V1とV2)と，PC-8801mkⅡMR用として，開発された，N88-日本語BASICとを備えています．

右の表で，PC-8800シリーズの各機種の持つBASICをまとめます．

BASICの種類＼PC-8800シリーズ	PC-8801 mkⅡMR	PC-8801 mkⅡFR	PC-8801 mkⅡSR	PC-8801 mkⅡ	PC-8801
N-BASIC			○	○	○
N88-BASIC V1	○	○	○	○	○
N88-BASIC V2	○	○	○		
N88-日本語BASIC (PC-8801mkⅡFR添付)		○	△		
N88-日本語BASIC (PC-8801mkⅡMR添付)	○	△*	△*		

○：標準実装しているもの
△：オプション
＊ 別売りの，ミニフロッピーディスクユニット(1Mバイトタイプ) PC-8831-MWと増設RAMが必要です．

資料1.2 各BASICの特長

それぞれのBASICは，基本的な命令や関数はほとんど同じですが，ハードウェアの機能に合わせて，グラフィックスやサウンドなどの機能に違いがあります．ここでは，それぞれのBASICの特長をまとめておきます．

N-BASIC

PC-8001の持つBASIC．PC-8001はグラフィック専用のRAMを持っていなかったため，テキスト画面をグラフィックス用に兼用していました．分解能は160×100ドットです．

N_{88}-BASIC V1

PC-8801, PC-8801MKⅡのメインBASIC. グラフィックRAMを活用して，多彩なグラフィック命令を持ちます.

N_{88}-BASIC V2

PC-8801MKⅡSRのメインBASIC. N_{88}-BASIC V1に加えて，アナログパレット機能や，FM音源によるサウンド機能を持ちます. また，処理速度の高速化を図っています.

N_{88}-日本語BASIC(PC-8801MKⅡFR添付)

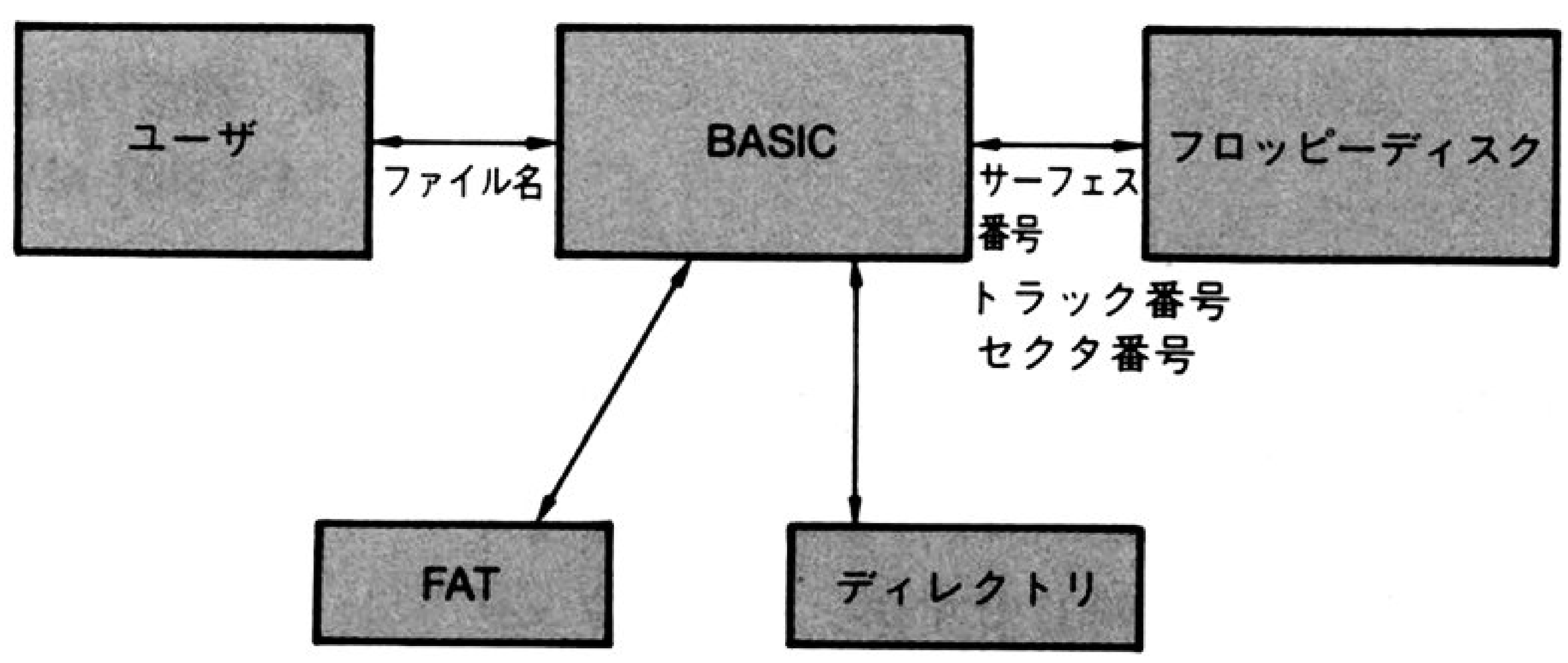
N_{88}-BASIC V2に，さらに日本語処理機能を加えたBASICです.

N_{88}-日本語BASIC(PC-8801MKⅡMR添付)

PC-8801MKⅡFRに添付されているN_{88}-日本語BASICでのかな漢字変換は熟語単位で行われますが，PC-8801MKⅡMRに添付されているN_{88}-日本語BASICは，文法処理を行いワープロ並みの文節変換が可能となります.

資料2. フロッピーディスクとファイルの構造

N_{88}-BASIC DISK version/N_{88}-日本語BASICにおけるフロッピーディスクのファイル構造について解説します.



BASICでディスクファイルを扱うときは，すべてファイル名で行います. したがって，フロッピーディスクを直接管理するサーフェス番号，トラック番号，セクタ番号を使う必要はほとんどありません. なぜならば，ファイル名を実際のフロッピーディスクのサーフェス番号，トラック番号，セクタ番号に変換する仕事はBASICが自動的に行ってくれるからです.

したがって，BASICでファイルを扱われる方は以下の説明を読む必要はありません. DSKI$関数，DSKO$文や機械語モニタを使う必要のある方のみお読みください.

資料2.1 フロッピーディスク

N_{88}-BASIC DISK version/N_{88}-日本語BASICで通常使用するフロッピーディスクは次の2種類です．ただし，N_{88}-日本語BASICのシステムディスクは，5.25インチのミニフロッピーディスクに限られます．

(1) **5.25インチミニフロッピーディスク（両面高密度）***

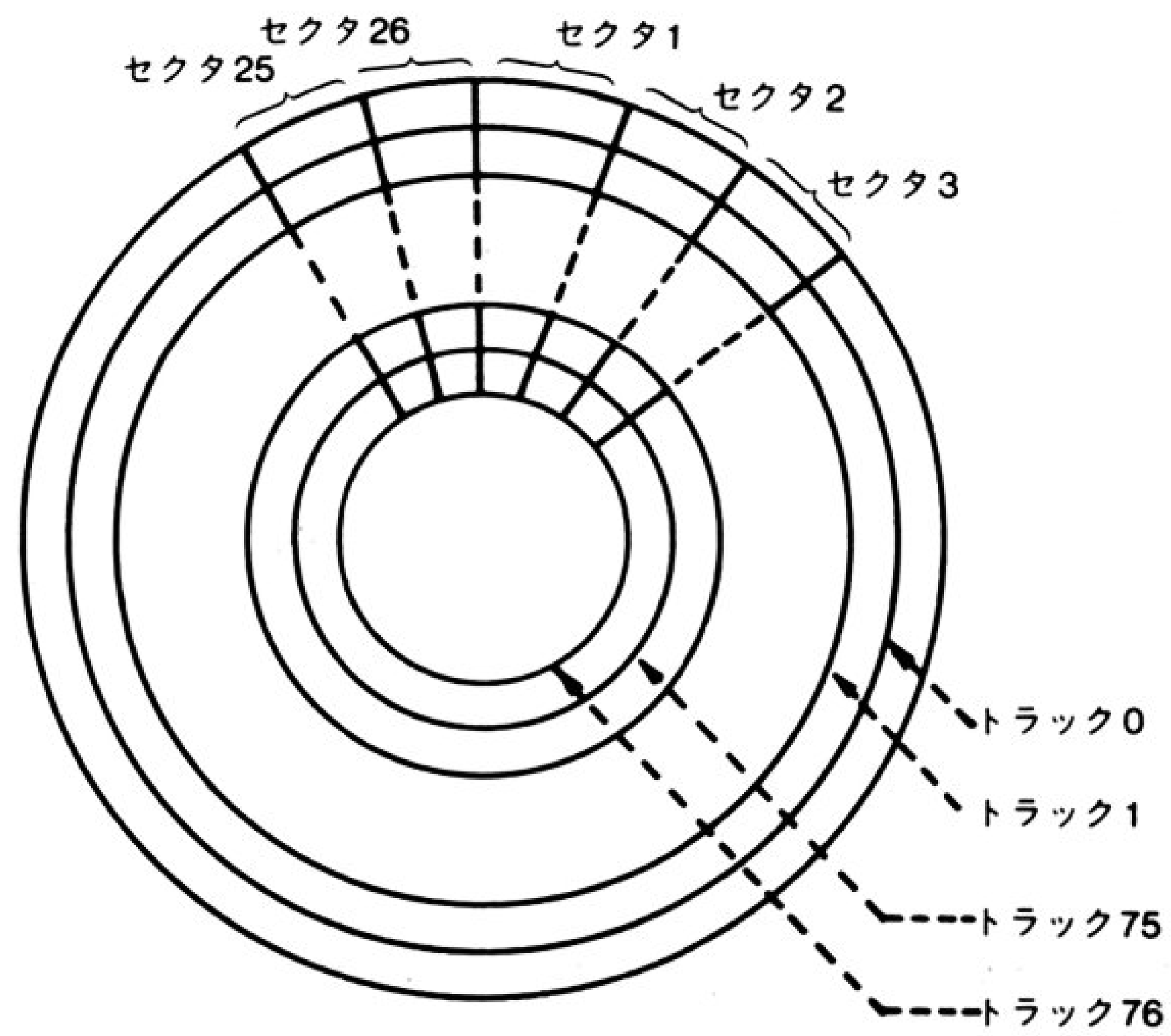
(2) **8インチフロッピーディスク（両面倍密度）****

* 2HDと呼ばれます．

** 2Dと呼ばれます．

フロッピーディスクにはメモリと同じように番地が付いています．フロッピーディスクの番地指定にはサーフェス番号，トラック番号，セクタ番号を使用します．サーフェス番号は，両面倍密度のフロッピーディスクを使用するときに必要です．「0」を指定すると表面（ラベルの貼ってない方面），「1」を指定すると裏面になります．フロッピーディスクの最小読み書き単位はセクタで，1セクタは256バイトからなります（サーフェス0のトラック0だけは，1セクタ128バイトです．この部分は，BASICでは使用できません）．

BASICで使用する2種類のフロッピーディスクの構成は，右のようになっています．2種類のフロッピーディスクは，全く同じ構成になっています．



8インチ両面倍密度フロッピーディスク
5.25インチ両面高密度フロッピーディスク

サーフェス番号	トラック数	トラック番号	セクタ数	セクタ番号
0か1	片面77 (154)	0～76	26	1～26

資料2.2 トラックの割り当て

各トラックの割り当ては次のようになっています.

システムディスクとはディスクコードを含んだフロッピーディスクのことで，N_{88}-BASIC DISK version，およびN_{88}-日本語BASICを起動させる際に必要なものです．ディスクコードとは，BASICの拡張部分(フロッピーディスク関係の命令のサポート)のことです．それとは逆にディスクコードを含まないフロッピーディスクのことをデータディスクと呼びます．データディスクではN_{88}-BASIC DISK version，およびN_{88}-日本語BASICを起動させることはできません.

プログラムやデータは，システムディスクとデータディスクの両方に記憶させることができます.

(1) 5.25インチ両面高密度ミニフロッピーディスク

N_{88}-BASICシステムディスク

サーフェス	トラック		セクタ		内容
	10進	16進	10進	16進	
0	0	0H	1～2	1H～2H	IPL
	0	0H	3～26	3H～1AH	未使用
	1	1H	1～2	1H～2H	IPL
	1	1H	3～26	3H～1AH	ディスクコード
	2	2H	すべて	すべて	ディスクコード
	3～34	3H～22H	すべて	すべて	ユーザ領域
	35	23H	1～22	1H～16H	ディレクトリ
	35	23H	23	17H	ID
	35	23H	24～26	18H～1AH	FAT
	36～76	24H～4CH	すべて	すべて	ユーザ領域
1	0	0H	すべて	すべて	未使用
	1	1H	すべて	すべて	ディスクコード
	2～76	2H～4CH	すべて	すべて	ユーザ領域

N_{88}-日本語BASICシステムディスク

サーフェス	トラック		セクタ		内容
	10進	16進	10進	16進	
0	0	0H	1～2	1H～2H	IPL
	0	0H	3～26	3H～1AH	未使用
	1～5	1H～5H	すべて	すべて	ディスクコード
	6～31	6H～1FH	すべて	すべて	辞書
	32～34	20H～22H	すべて	すべて	ユーザ領域
	35	23H	1～22	1H～16H	ディレクトリ
	35	23H	23	17H	ID
	35	23H	24～26	18H～1AH	FAT
	36～76	24H～4CH	すべて	すべて	ユーザ領域
1	0	0H	すべて	すべて	未使用
	1～5	1H～5H	すべて	すべて	ディスクコード
	6～31	6H～1FH	すべて	すべて	辞書
	32～76	20H～4CH	すべて	すべて	ユーザ領域

データディスク

サーフェス	トラック		セクタ		内容
	10進	16進	10進	16進	
0	0	0H	すべて	すべて	未使用
	1～34	1H～22H	すべて	すべて	ユーザ領域
	35	23H	1～22	1H～16H	ディレクトリ
	35	23H	23	17H	ID
	35	23H	24～26	18H～1AH	FAT
	36～76	24H～4CH	すべて	すべて	ユーザ領域
1	0	0	すべて	すべて	未使用
	1～76	1H～4CH	すべて	すべて	ユーザ領域

(2) 8インチ両面倍密度フロッピーディスク

N_{88}-BASICシステムディスク

サーフェス	トラック		セクタ		内容
	10進	16進	10進	16進	
0	0	0H	すべて	すべて	未使用
	1	1H	1	1H	IPL
	1	1H	2～26	2H～1AH	ディスクコード
	2	2H	すべて	すべて	ディスクコード
	3～34	3H～22H	すべて	すべて	ユーザ領域
	35	23H	1～22	1H～16H	ディレクトリ
	35	23H	23	17H	ID
	35	23H	24～26	18H～1AH	FAT
	36～76	24H～4CH	すべて	すべて	ユーザ領域
1	0	0H	すべて	すべて	未使用
	1	1H	すべて	すべて	ディスクコード
	2～76	2H～4CH	すべて	すべて	ユーザ領域

データディスク

サーフェス	トラック		セクタ	内容
	10進	16進		
0	0	0H	すべて	未使用
	1～34	1H～22H	すべて	ユーザ領域
	35	23H	すべて	システムディスクと同じ
	36～76	24H～4CH	すべて	ユーザ領域
1	0	0H	すべて	未使用
	1～76	1H～4CH	すべて	ユーザ領域

資料2.3 クラスタ

フロッピーディスクで読み書き可能な最小単位はセクタですが，BASICが管理する最小単位はクラスタと呼びます．

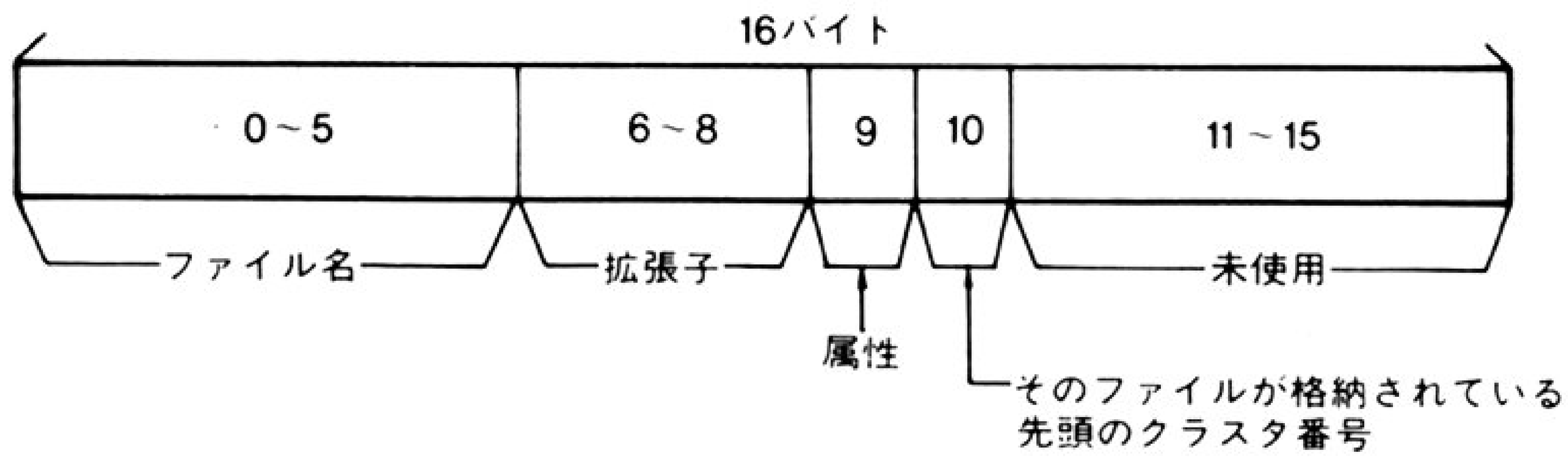
セクタをひとまとめにして，1クラスタとします．クラスタとサーフェス，トラック，セクタの関係は右の表のようになっています．

クラスタ		サーフェス		トラック		セクタ	
10進	16進	10進	16進	10進	16進	10進	16進
0	0H	0	0H	0	0H	1～26	1H～1AH
1	1H	1	1H	0	0H	1～26	1H～1AH
2	2H	0	0H	1	1H	1～26	1H～1AH
3	3H	1	1H	1	1H	1～26	1H～1AH
4	4H	0	0H	2	2H	1～26	1H～1AH
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
152	98H	0	0H	76	4CH	1～26	1H～1AH
153	99H	1	1H	76	4CH	1～26	1H～1AH

資料2.4 ディレクトリ，FAT

ユーザから指定されたファイル名を，フロッピーディスク上のアドレス(サーフェス番号，トラック番号，セクタ番号)に変換するために，BASICでは2つの表を管理しています．ディレクトリとFAT(File Allocation Tableの略)がそれです．

ディレクトリは，そのディスクに格納されているファイルのファイル名，属性，格納されている先頭の場所の表です．各ディレクトリは16バイトからなり，右のような内容を持っています．



バイト		内容
10進	16進	
0～5	0H～5H	ファイル名
6～8	6H～8H	拡張子
9	9H	属性
10	AH	そのファイルが格納されている先頭のクラスタ番号
11～15	BH～FH	未使用

ファイル名の先頭バイト(0バイト目)の内容がFF(16進数)のとき，そのディレクトリが使用されていないことを示します．00のとき，削除(KILL)されたファイルであることを示します．

属性は右のようになっています．

値(16進数)	意味
0H	アスキー形式
1H	機械語ファイル
10H	アスキー形式，書き込み禁止
40H	アスキー形式，リードアフターライト
80H	非アスキー形式
90H	非アスキー形式，書き込み禁止
A0H	非アスキー形式，暗号化
C0H	非アスキー形式，リードアフターライト

FATは各クラスタの使用状態を示す表です．FATは3セクタにわたって書き込まれており，3セクタとも全く同じ内容が書かれています．フロッピーディスクをアクセスしたときに，FATの内容(3セクタ)がメモリにコピーされます．1クラスタにつき1バイトで，そのクラスタの使用状態を表します．各セクタの0H～99Hバイト目がクラスタ番号0H～99Hにそれぞれ対応しています．そして，各バイトの持っている値が各クラスタの使用状態を示しています．

表の説明で「連続したクラスタ」といいましたが，これは物理的(地理的)に連続したクラスタという意味ではありません．BASICで管理されるファイルは一般に，物理的にとびとびのクラスタをつないで構成されています．

バイトのデータ(16進)	クラスタの使用状態
0H～99H	使用中．連続したクラスタの一部であり，後続するクラスタを持つ．値が，後続するクラスタの番号を示している．
C1H～DAH	使用中．連続したクラスタの最後のクラスタであり，下5ビットの内容が，そのクラスタで実際に使われているセクタの数を表す．
FEH	予約済みのクラスタで，ファイルとして使うことはできない．(ディスクコード，IPL，ディレクトリ，FAT自身を含むクラスタがこれである．)
FFH	未使用

資料2.5 ID

IDには1セクタ(=256バイト)が割り当てられています．各バイトは右のような意味を持っています．

属性	ファイル数	BASICテキスト
1バイト	1バイト	254バイト

最初の1バイトは，そのフロッピーディスク全体の属性を示しています．値の意味は右のとおりです．

値(16進数)	意　　味
0H	読み書き可
10H	書き込み禁止
40H	リードアフターライト

通常のシステムディスクでは，〈ファイル数〉として255(16進数ではFF)が指定されています．この場合，システム起動時にキーボードからファイル数を入力する必要があります．このファイル数に0～15を指定すると，BASICをスタートさせるときに

How many files(0-15)?

を表示せずに，自動的にそのファイル数が設定されます．

ファイル数の設定が終わると(ファイル数を設定せずに，「How many files(0-15)?」に対してキーボードから入力を行った場合でも)，次のBASICテキストの内容が自動的に実行されます．つまり，BASICをスタートさせる場合，システムディスクのIDに書かれているBASICテキストの内容は，常に実行されます．

また，ディスクユーティリティにより年の設定を行った場合にも，このBASICテキストのエリアが使われます．

IDの通常の状態は，ファイル数が255(FFH)，BASICテキストがすべて0か32(0Hか20H)になっています．

例

```
00 05 72 75 6E 22 41 00…………00
(r)(u)(n)(")(A)(16進数)
```

この例のようにIDセクタの内容が書かれている場合，自動的に5個のバッファが用意され，"A"というプログラムが走り出します．

IDの書き換えを行いたい場合は，システムディスクに入っているディスクユーティリティを使ってください．

資料2.6 そのほかのフォーマットのフロッピーディスク

資料2.1～2.5までででは，次の2つのタイプのフロッピーディスクの構造に関して述べました，

(1) 5.25インチミニフロッピーディスク（両面高密度）*

(2) 8インチフロッピーディスク（両面倍密度）**

* 一般に2HDと呼ばれています．

** 一般に2Dと呼ばれています．

しかし，フロッピーディスクのフォーマットには，上の2つのほかにもいくつか種類があり，PC-8800シリーズやPC-9800シリーズ用の市販のソフトウェアや，BASIC DISK versionで使われています．ここでは，次の3つのタイプのフロッピーディスクに関して補足説明をします．

(3) 5.25インチミニフロッピーディスク（両面倍密度）**

(4) 5.25インチミニフロッピーディスク（片面倍密度）***

(5) 5.25インチミニフロッピーディスク（両面倍密度倍トラック）****

** 一般に2Dと呼ばれています．

*** 一般に1Dと呼ばれています．

**** 一般に2DDと呼ばれています．

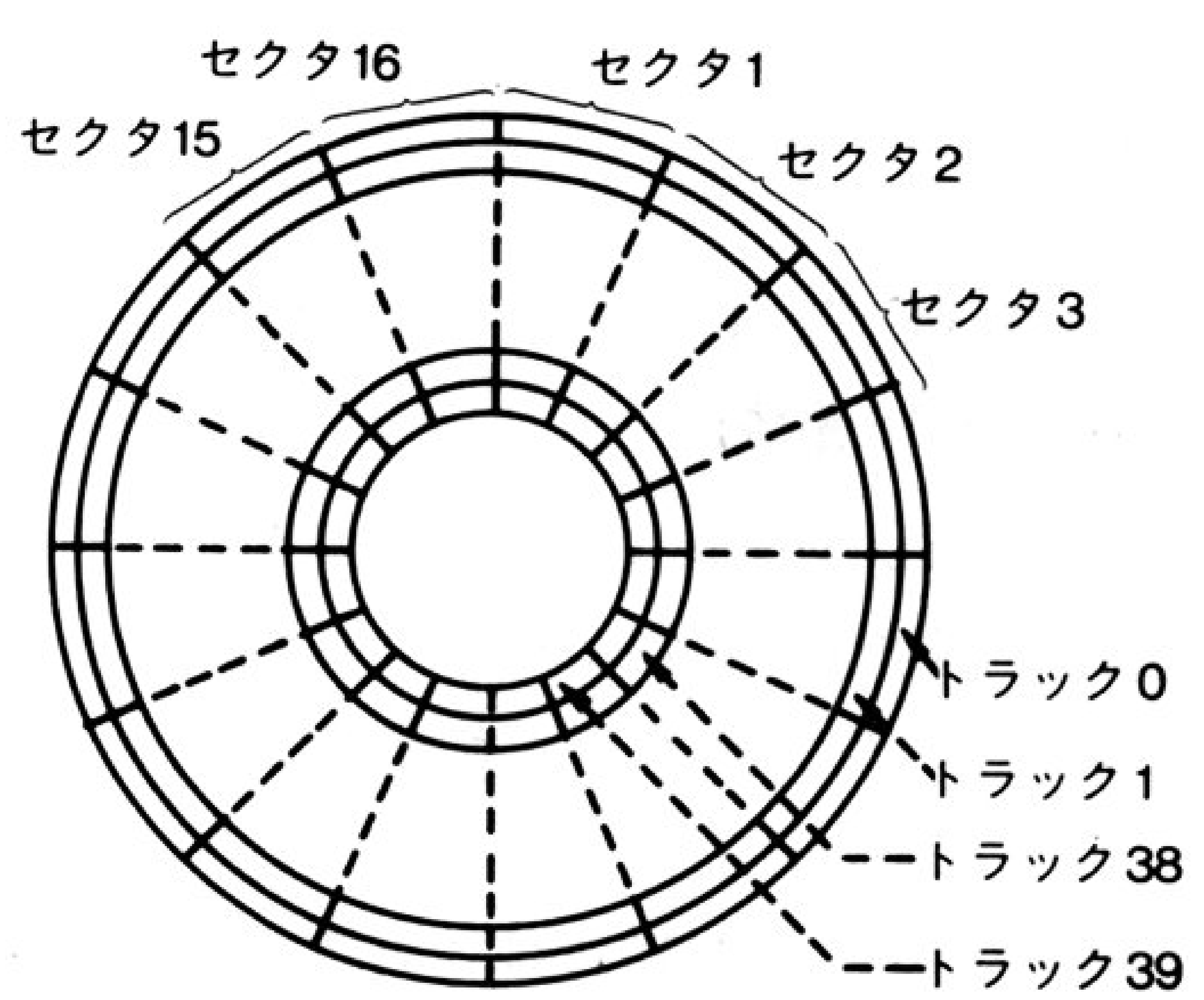
(3) 5.25インチミニフロッピーディスク（両面倍密度）

PC-8000シリーズ，PC-8800シリーズの従来機BASIC DISK versionでは，このフォーマットを用いています．

PC-8801MKⅡMRでも，ディスクユーティリティの「システムディスクの作成」によって両面倍密度のN_{88}-BASICシステムディスクを作り，これでN_{88}-BASIC DISK versionをスタートすれば，従来機で作ったファイルを使用することができます．

右に，5.25インチミニフロッピーディスク（両面倍密度）の構成と，「システムディスクの作成」によって作られるディスクの内容をあげます．

注意：左記のようにして，PC-8801MKⅡMRで，両面倍密度のファイルを読み書きすることができます．ただし，このようにして作ったファイルを従来機で読む場合，正常に動作しないことがあります．



5.25インチ両面倍密度ミニフロッピーディスク

サーフェス番号	トラック数	トラック番号	セクタ数	セクタ番号
0か1	片面40（80）	0～39	16	1～16

N88-BASICシステムディスク

サーフェス	トラック		セクタ		内容
	10進	16進	10進	16進	
0	0	0H	1	1H	IPL
	0	0H	2～16	2H～10H	ディスクコード
	1	1H	すべて	すべて	ディスクコード
	2～39	2H～27H	すべて	すべて	ユーザ領域
1	0～1	0H～1H	すべて	すべて	ディスクコード
	2～17	2H～11H	すべて	すべて	ユーザ領域
	18	12H	1～12	1H～CH	ディレクトリ
	18	12H	13	DH	ID
	18	12H	14～16	EH～10H	FAT
	19～39	13H～27H	すべて	すべて	ユーザ領域

クラスタの割り当て

クラスタ		サーフェス		トラック		セクタ	
10進	16進	10進	16進	10進	16進	10進	16進
0	0H	0	0H	0	0H	1～8	1H～8H
1	1H	0	0H	0	0H	9～16	9H～10H
2	2H	1	1H	0	0H	1～8	1H～8H
3	3H	1	1H	0	0H	9～16	9H～10H
4	4H	0	0H	1	1H	1～8	1H～8H
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
158	9EH	1	1H	39	27H	1～8	1H～8H
159	9FH	1	1H	39	27H	9～16	9H～10H

(4) 5.25インチミニフロッピーディスク（片面倍密度）

(5) 5.25インチミニフロッピーディスク（両面倍密度トラック）

上の(4)は，PC-8801, PC-8001などに片面用ドライブPC-8031-1Wを接続したシステムで用いていたフォーマットです．(5)は，PC-9800シリーズで使用しているフォーマットです．

これらのフォーマットで作られているファイルは，ディスクユーティリティの「ファイル転送」を使って，PC-8801MKⅡMRで使用できるフォーマットに転送してから使います．

資料3. 音の原理

資料3.1 FM音源

PC-8801mkIIMRでは，FM音源による音楽演奏ができます．内蔵されているシンセサイザIC(OPN)を制御して思いどおりの音を出すためには，まずFM音源の原理を知る必要があります．

1. FM音源とは

FM音源はデジタル音源が基礎となっています．デジタル音源は音を何もない状態から"発生"させるのではなく，コンピュータに使われているのと同じような記憶回路から，いろんなタイミングで音を読み出し"再現"する方式をとっています．

2. "波形＋時間"が"音"になる

この記憶回路にはサイン波だけが，"1"と"0"だけでデジタル符号化されて入っています．

ここにひとつの記憶回路があるとします．その中にはサイン波が，図1のように10個の電圧として蓄えられています．a～jの電圧が四捨五入され，二進化されて入っています．大切なのは最も下側の欄で，サイン波はすべてこのような"1"と"0"に置き換わっているということです．しかしこれは説明のための単なる一例で，実際の楽器では，サイン波は必ずしも10個の電圧に分割されているわけではありません(もっと多くの電圧に分割されている)し，電圧の値もこうとは限りません．

このような形で記憶回路に入っているのは，"サイン波のもと"でしかありません．というのは，記憶回路にはサイン波を構成する電圧が順番どおりに入っているだけで，それらをどういったタイミングで並べるかのデータは入っていないからです．

aとb，bとcの間を，どのくらいの時間にするか(1マイクロセカンドなのか1秒なのか……)のデータは全然入っておらず，それが決まらなければ"音"にはなりません．aとb, bとcといった各電圧の間の"時間"を決めて，初めて"音"になります．

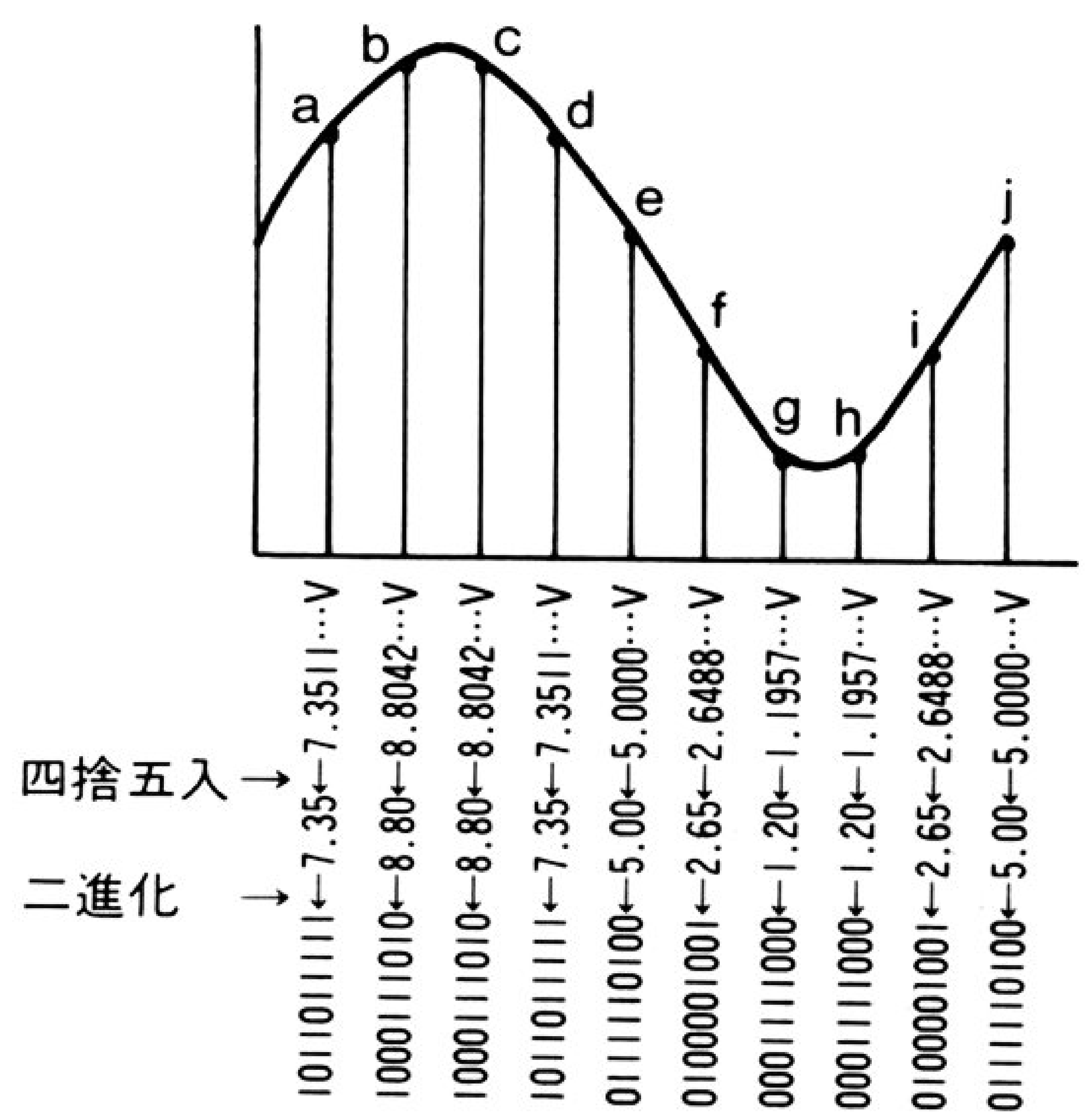


図1

もしも"時間"が決まらなければ，サイン波の１サクルの時間もわからないわけですから，周波数が決まりません．波形と周期（１サイクルの所要時間）が決まって，初めて"音"になるのです．

3．記憶回路に"時間"を与えると……

記憶回路に"時間"のデータを与えるにはどうしたらいいのでしょうか? 図２はノコギリ波です．今かりに，このノコギリ波が0Vから上昇し，10Vでストンともとに戻るような振幅だったとしてこれを記憶回路に与えます．そして，ノコギリ波の電圧が1Vのときに記憶回路のa.の電圧を，2Vならb.……としていき，10Vになったらj.の電圧を記憶回路から出力させるようにしておきます．このような約束にしておくと，ノコギリ波の電圧は直線的に上昇しますから，a.とb.，b.とc.といった隣り合った電圧が一定の時間間隔で出力に出てくることになります．図３がその様子です．

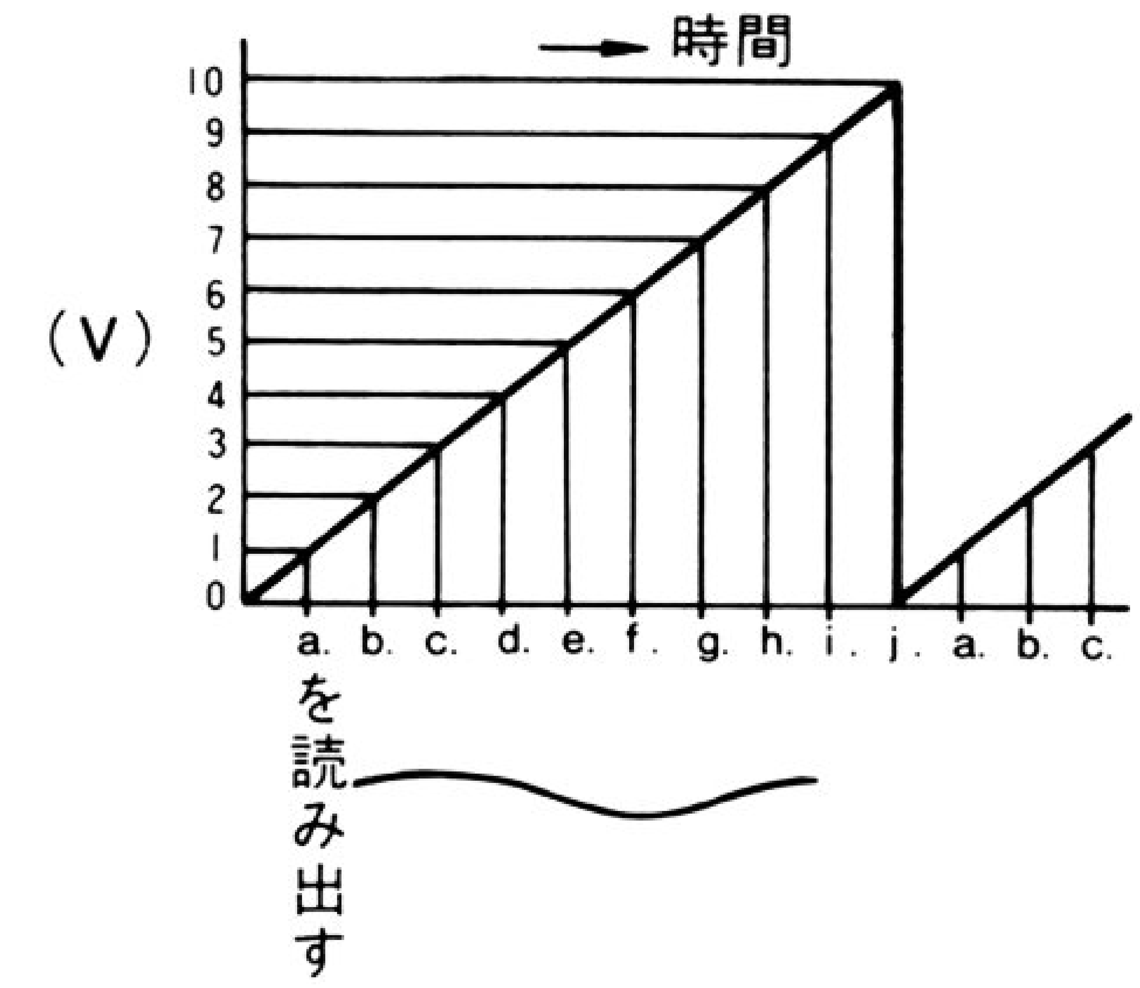


図２

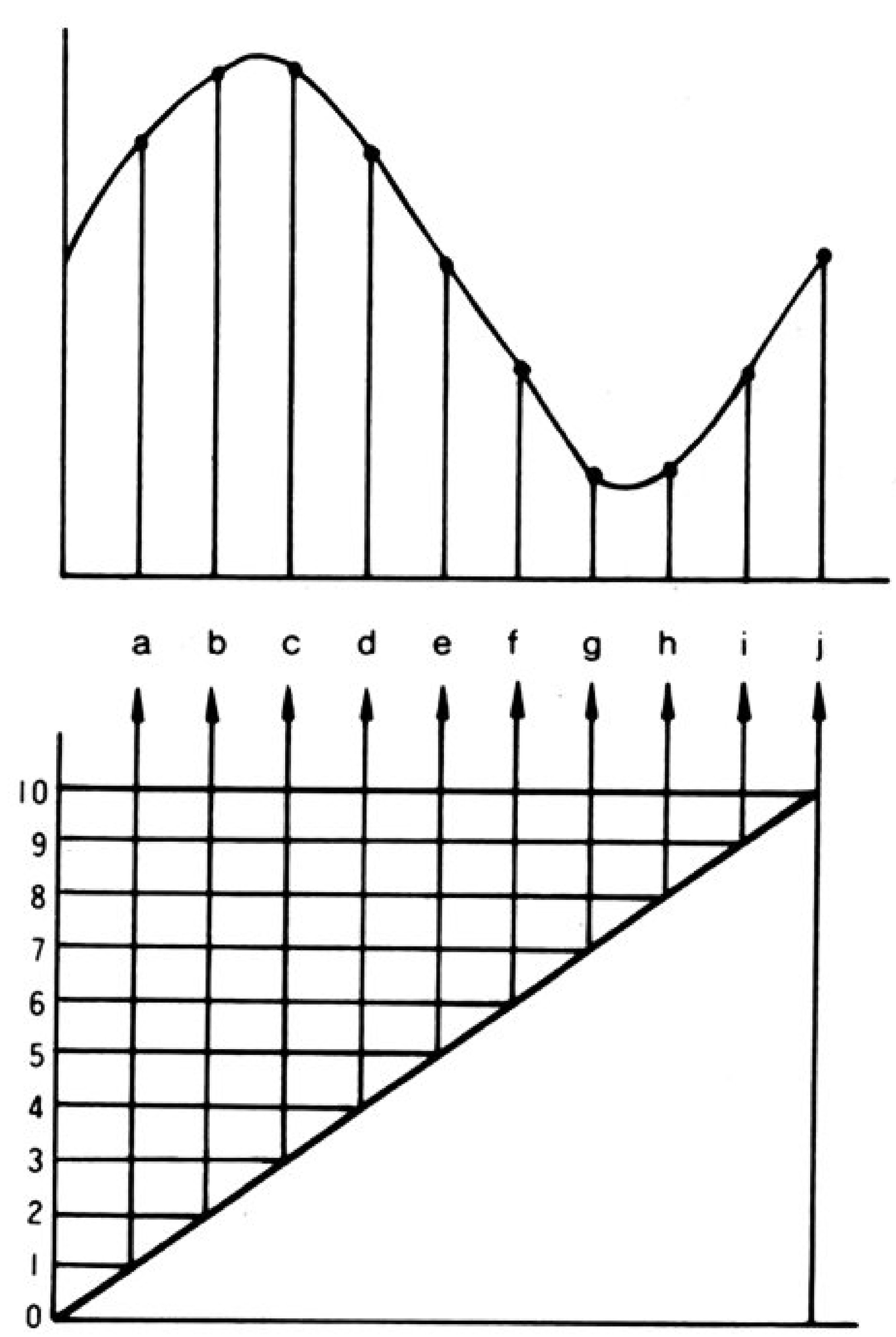


図３

この方法によって，記憶回路からきれいなサイン波が出てきます．そして，ここで大切なのは「読み出されたサイン波の周波数は，読み出しに使ったノコギリ波の周波数と同じである」ということです．図４を見てわかるように，ノコギリ波の１サイクルとサイン波の１サイクルは，常に一致しています．そして，１サイクルに要する時間が同じであれば周波数も同じになります

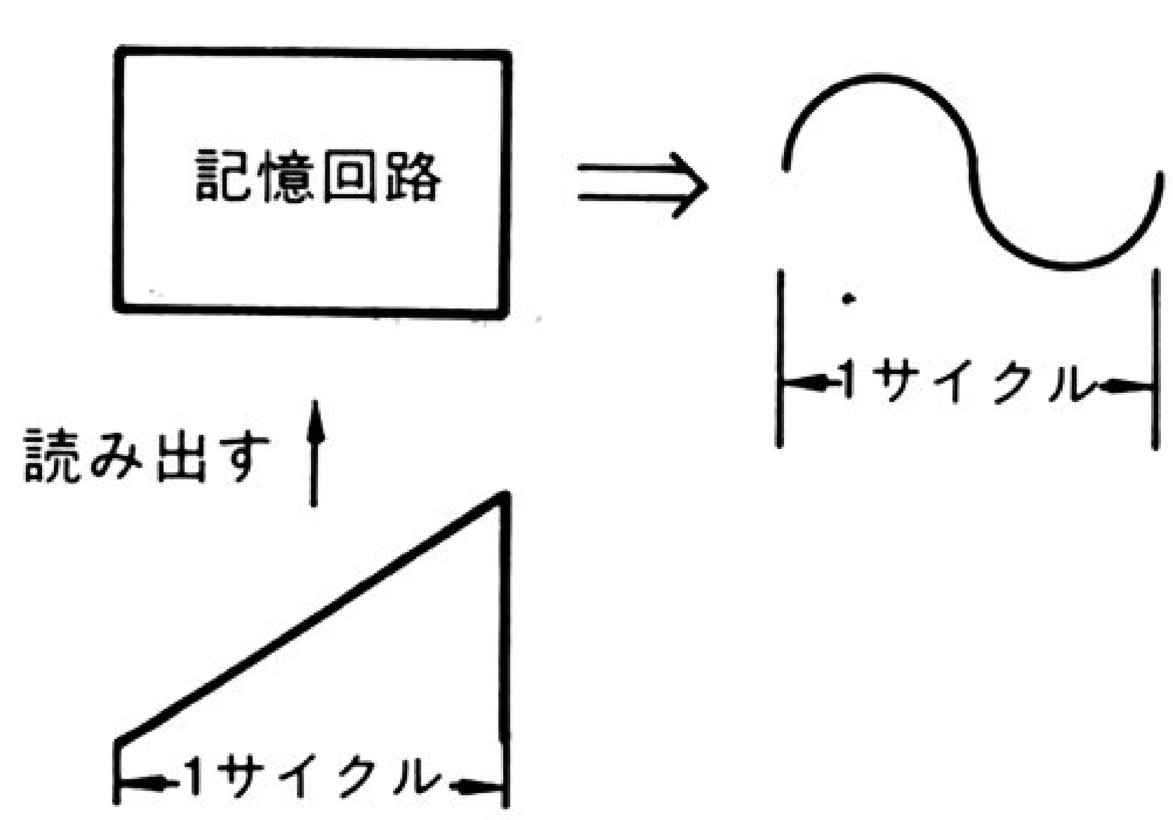


図４

（1サイクルが1ミリセカンドの波形は，これが連続している場合，1秒間に1000回繰り返されます．つまり1秒間に1000サイクルですから，これは1000Hzの周波数に当たります）．

これが，"デジタル音源"の基礎です．

4．記憶されているのは"サイン波"だけ

FM音源は，デジタル音源のひとつの発展型と考えられます．ですから，"記憶回路からノコギリ波を使ってサイン波を読み出す"のが第一歩となります．

図4をもう一度見てください．ここで，ノコギリ波を図5のように細工して記憶回路を読み出してみましょう．出てくるのはひしゃげたサイン波になってしまいます（図6）．どうしてでしょうか? 図7を見るとわかるように折れ曲ったノコギリ波は，2つのノコギリ波が合成されたもの，と考えれば簡単になります．

それでは図8のように，今度は逆に折ったらどうなるでしょうか? さっきとは逆にひしゃげたサイン波になります（図9）．

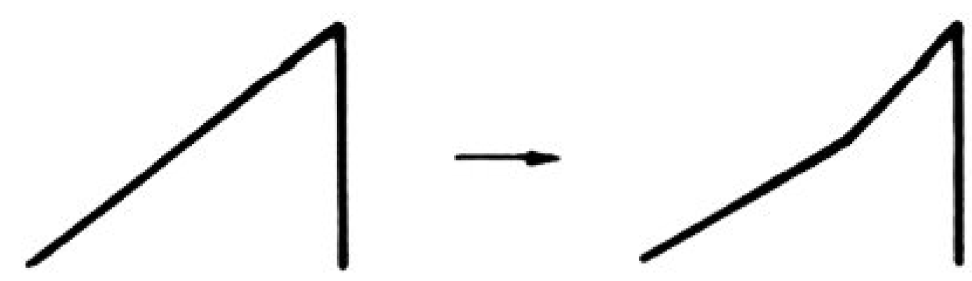


図5

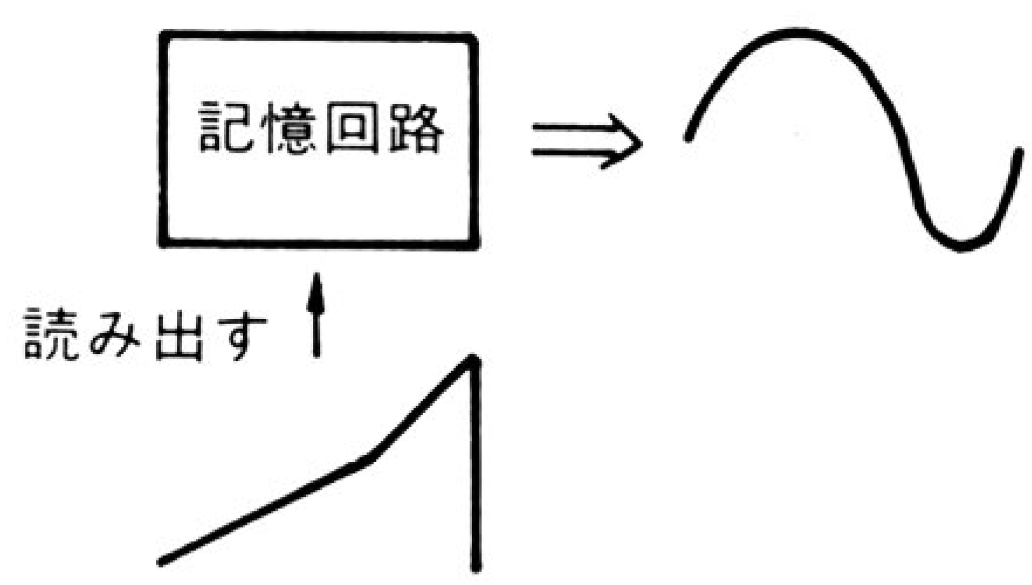


図6

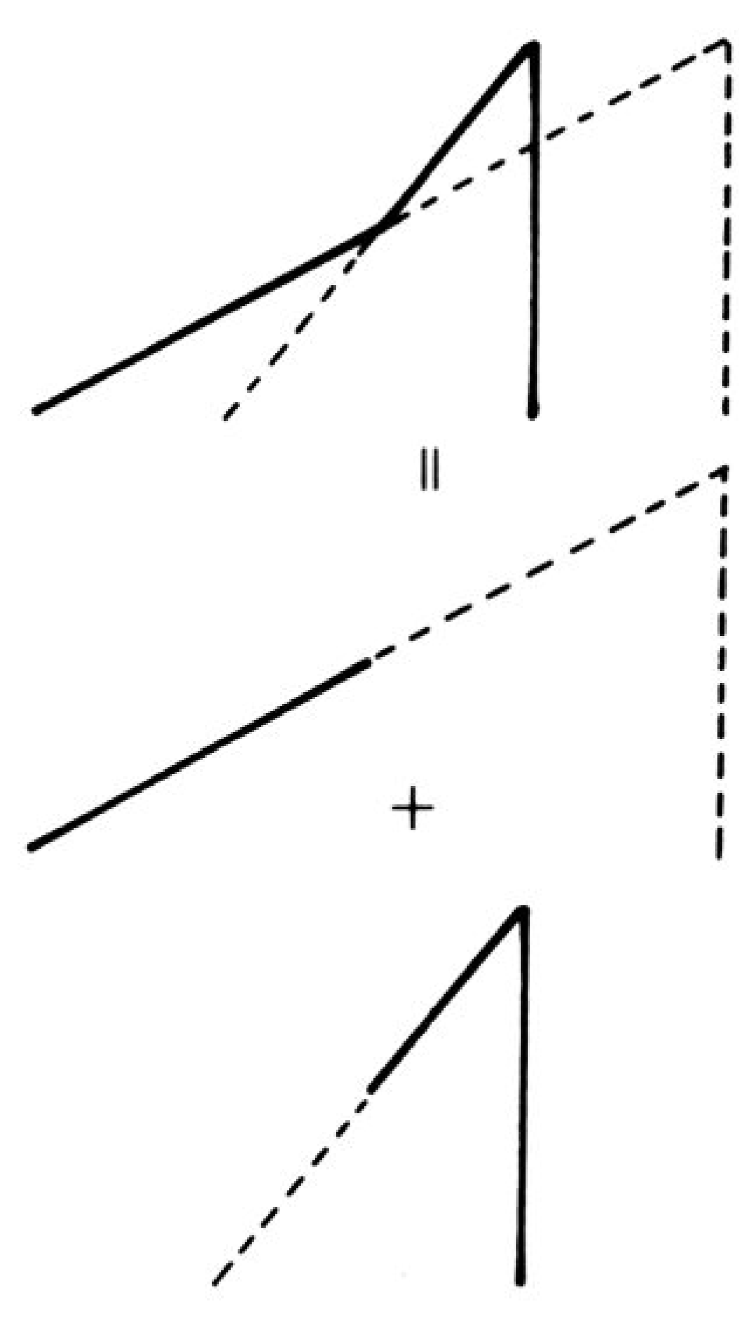


図7

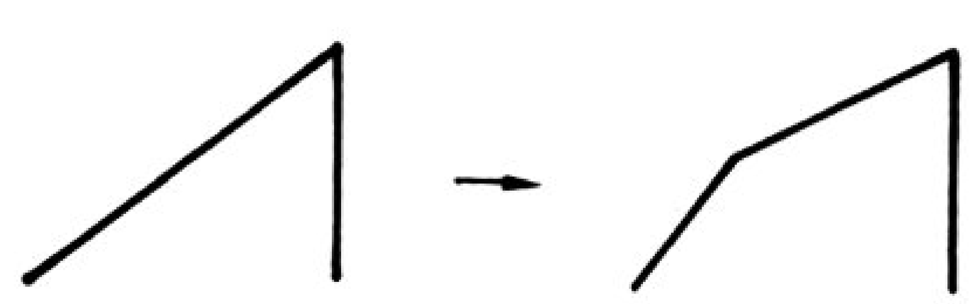


図8

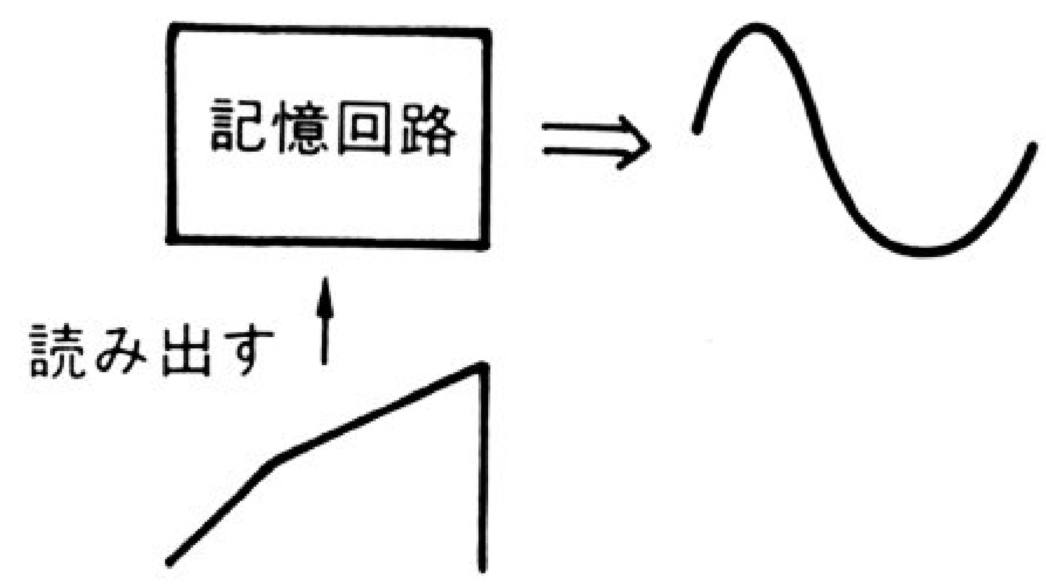


図 9

読み出しに使うのが，必ずしも正しいノコギリ波でなくてもいいのなら，たとえば図10のようなもっと複雑な折れ線の波形で読み出してみましょう．この折れ線で問題なのは㋺～㋩間です．なぜなら㋑～㋺，㋩～㋥間は，図11のように考えればそれぞれ"ノコギリ波の一部"と見なせますが，㋺～㋩では傾斜が逆です．つまり，逆向きのノコギリ波になってしまいます(図12)．

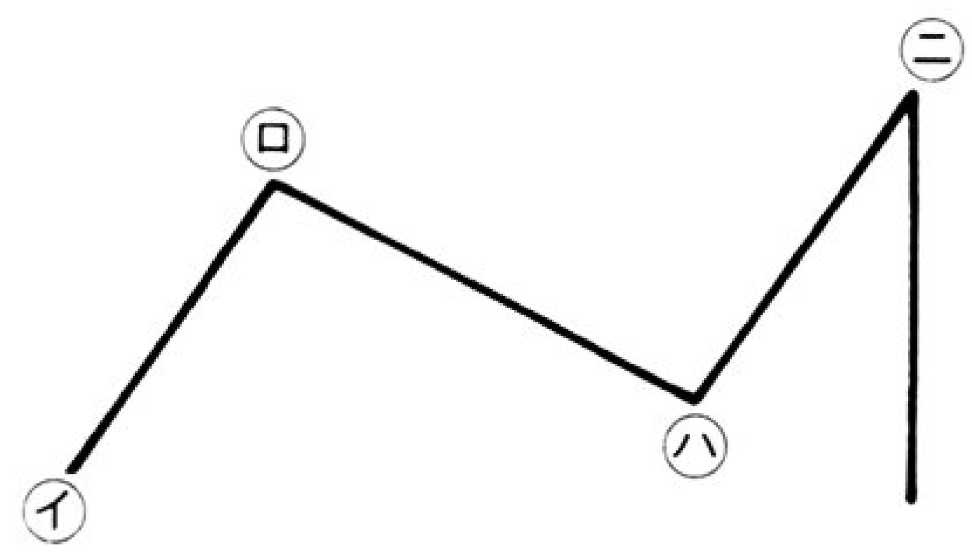


図10

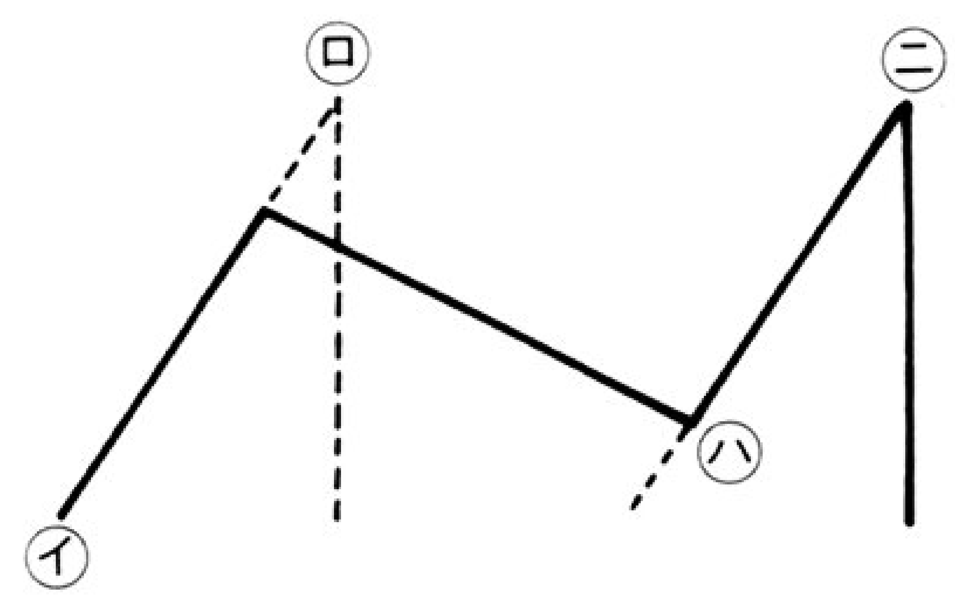


図11

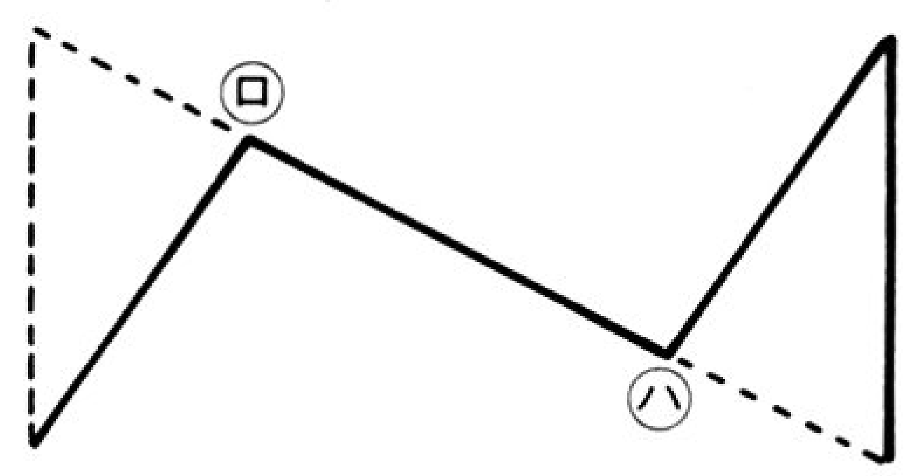


図12

そして図13が記憶回路から出てくる波形です．これはもうひしゃげすぎていてサイン波とは呼びがたいですね．しかし，これで"逆向きのノコギリ波"でも，読み出しには全く支障がないことがわかりました．どうしてでしょう？

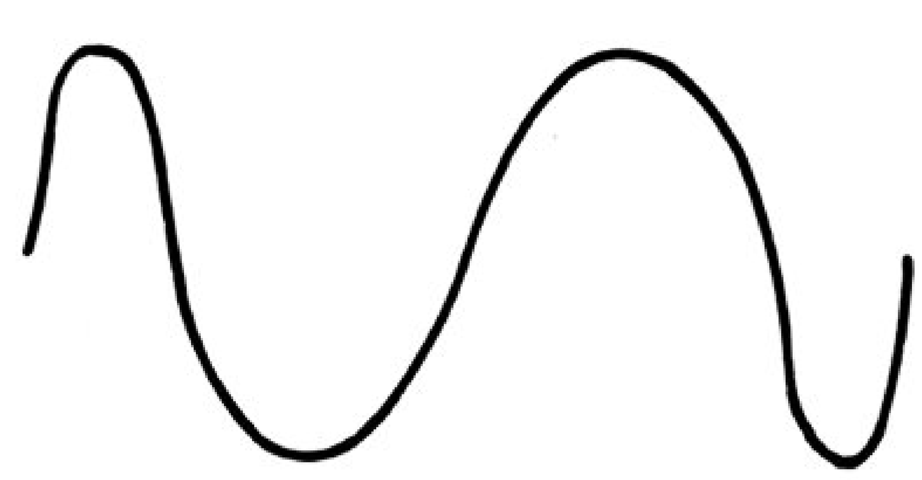

図13

図14で説明をしましょう．これはただ，読み出し用の波形を90度回転しただけのものです．まず㋑〜㋺では，記憶回路の中の電圧を順番に読み出します．ここまでは普通ですが，次に㋺〜㋩では，今度は記憶されている電圧を逆の順番で読みます．a. b. c.……と読んでいたものが，逆になるとg. f. e.……になります．そして，㋩〜㋥では再び正しい順番に戻ります．

このように記憶回路を読み出すのは，どんな波形(ノコギリ波でなくても)でもかまいません．そして，読み出しに使う波形によって，記憶回路に入っているサイン波は，いろんな形に変形されて出力に出てきます．これがFM方式の最も大切な基本です．

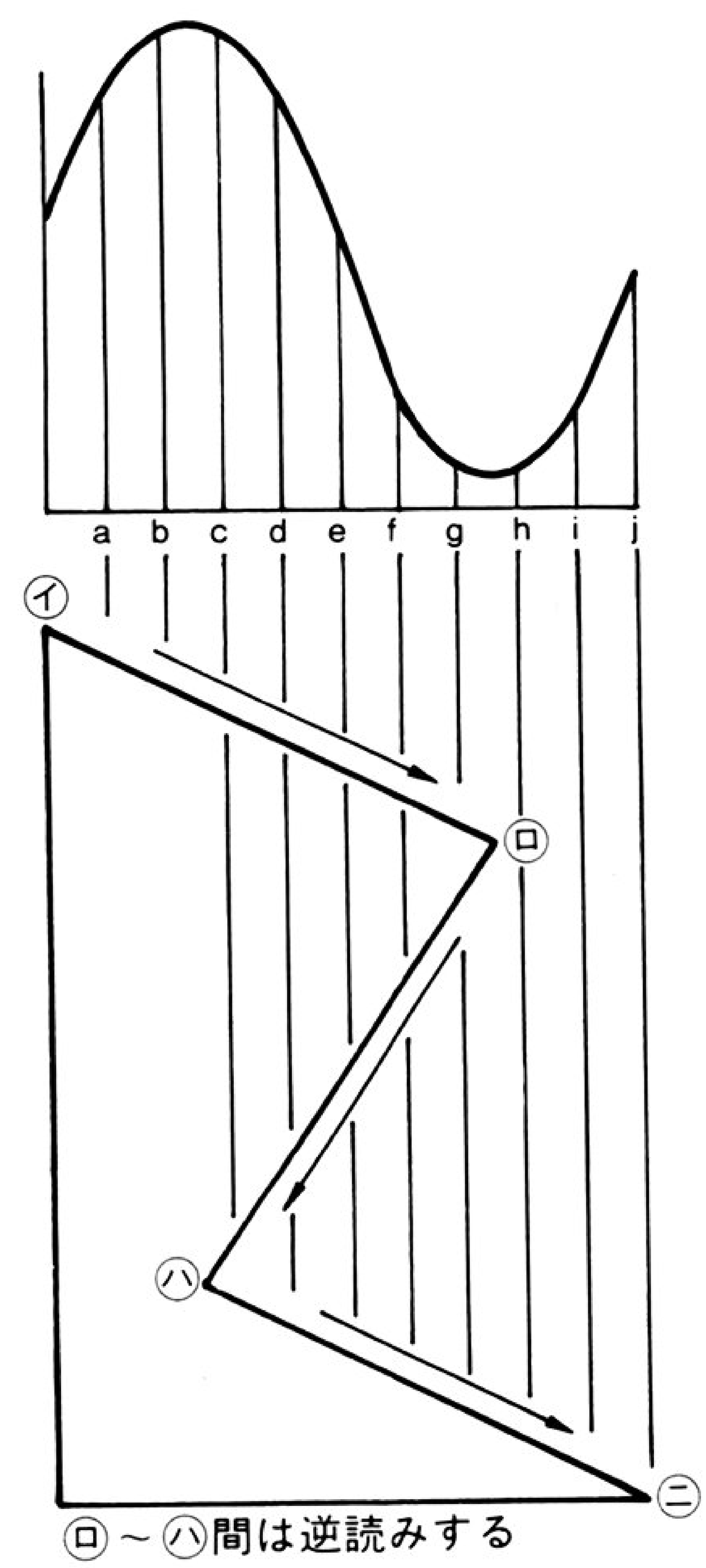


図14

5. "ノコギリ波＋サイン波"で読み出す

ところで図10をもう一度見てください．一見メチャクチャな波形ですが，実はこれは，ノコギリ波と三角波が足し合わされたものです．図15のように，シンプルな波形どうしでも．足し合わせる(以後"加算"という)ことによって，かなり複雑な波形にすることができます．

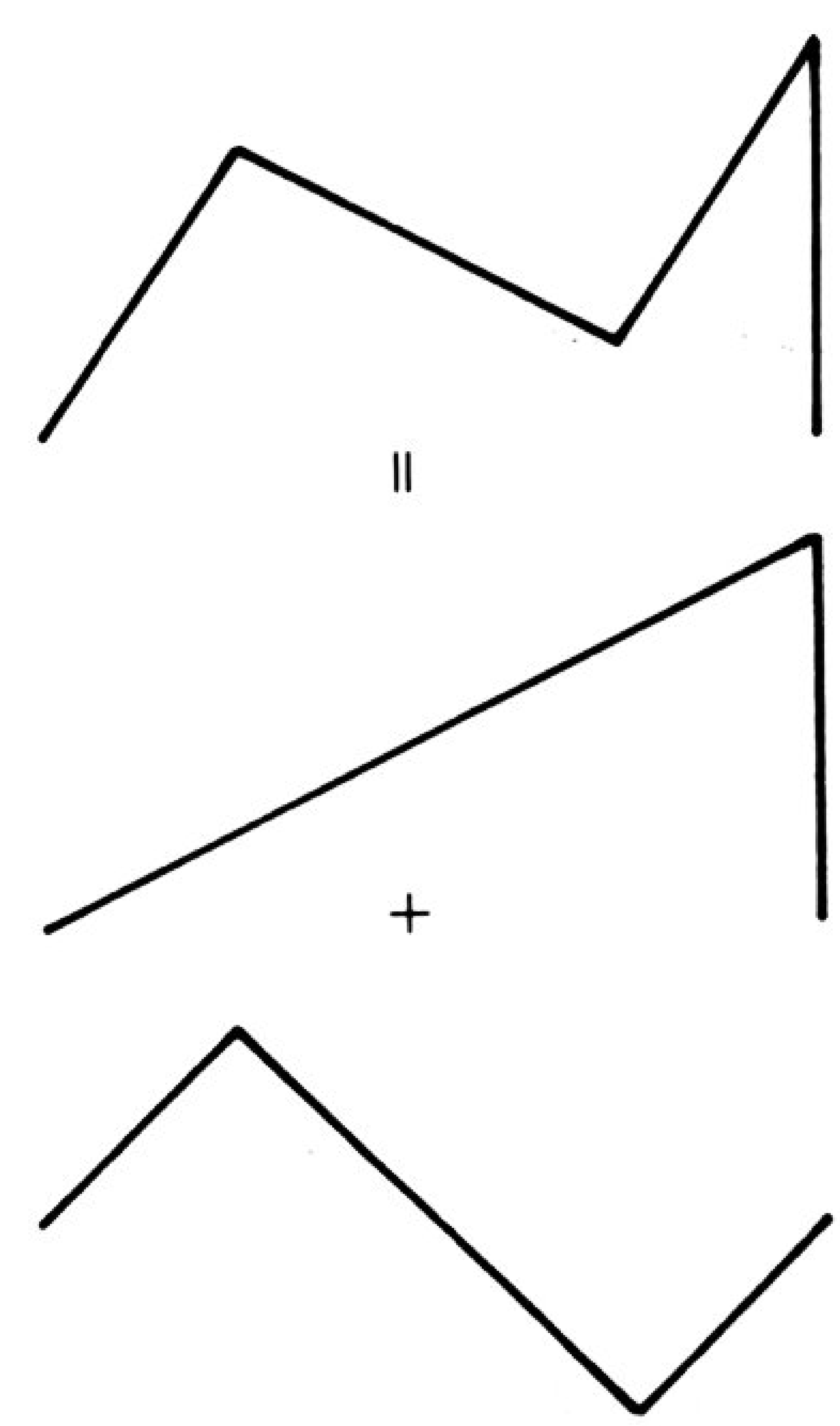


図15

三角波の代わりにサイン波をノコギリ波に加算したらどうでしょう．図16がそれです．そして，図17が記憶回路から出てくる波形です．

図16では，ノコギリ波にくらべてサイン波の振幅を小さくしておきました．今度は，サイン波の振幅を大きくしてみましょう．図18を見てください．これを使って記憶回路を読み出すと，図19の波形が得られます．図17と比べてみてください．同じ〔ノコギリ波＋サイン波〕で読み出しても，加算したサイン波の振幅を変えると，読み出される波形は違ったものになってきます．

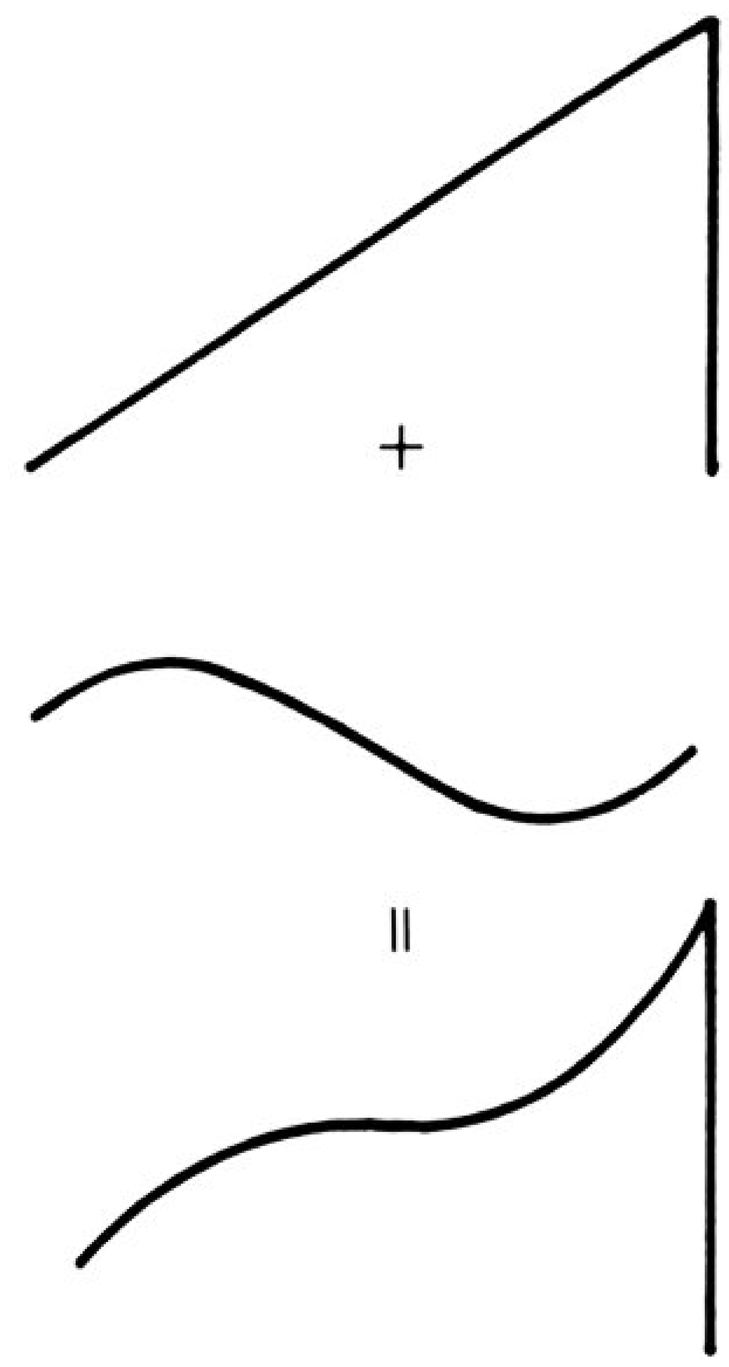


図16

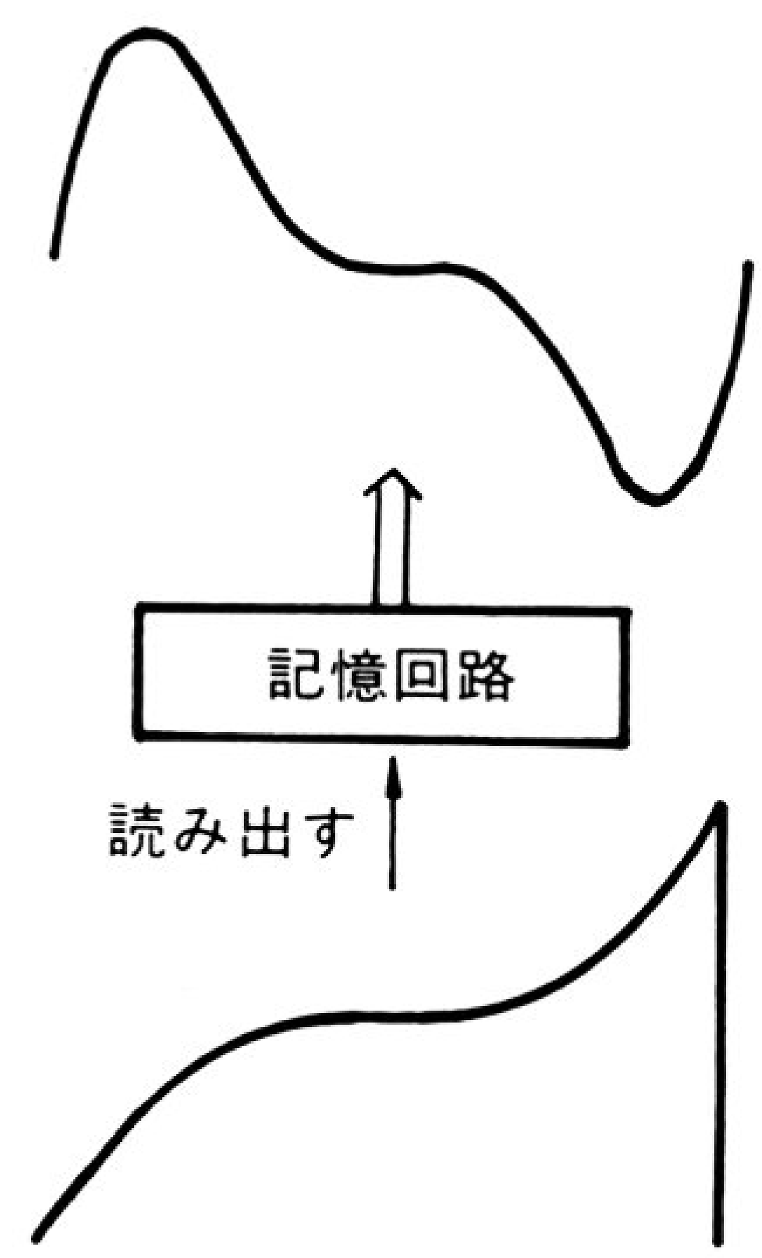


図17

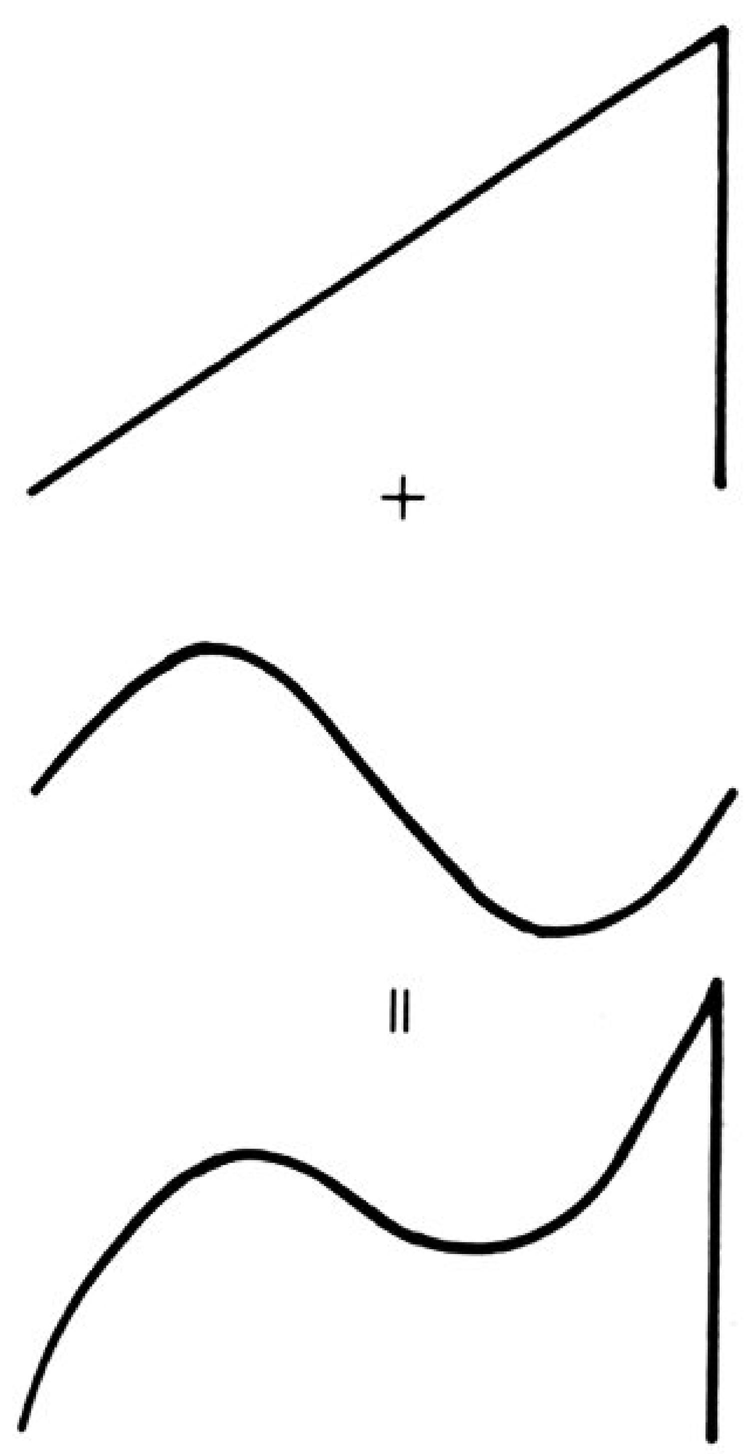


図18

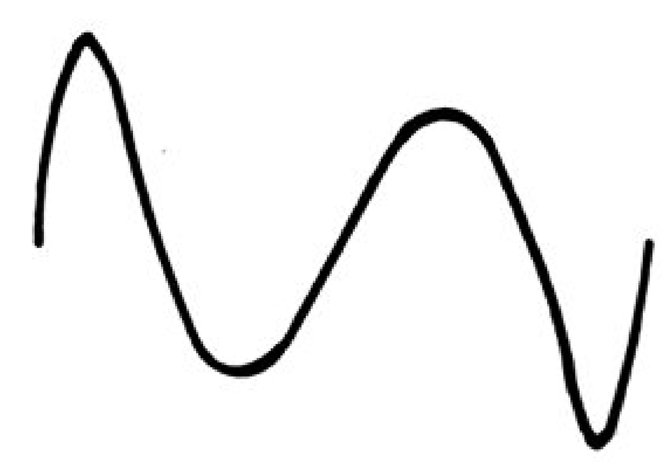


図19

6. ノコギリ波が"周波数"，サイン波が"音色"を決める

今まで書いた"読み出し"の例をよく見なおしてみると，サイン波の振幅を変えたり，あるいはサイン波の代わりに三角波を使った場合でも，読み出された波形の１サイクルの時間はすべてノコギリ波によって決定されています．ノコギリ波にどんな波形を加算しようとも，それによって読み出された波形の１サイクルは，ノコギリ波の１サイクルの時間と一致しています．これはデジタル方式の場合と全く同じで，ノコギリ波がすべての周波数を決めていることになります．

一方，ノコギリ波に加算れたサイン波はどんな役割を持っているのでしょうか？これらは，記憶回路から出てくる波形自体を決めています．つまり，音色を決定しています．

このようにノコギリ波とサイン波は，それぞれ決定すべき大きな役割を持っています．

7．2つの波形の周波数を変えると

これまでの例では，ノコギリ波にサイン波を加算するとき，両者の周波数比は1:1，つまり同じ周波数でした．そこで今度は，周波数比を変えて考えてみましょう．ノコギリ波は，出力の周波数を決定するものですからその周波数は変えません．変えるのはサイン波の周波数です．

まず，ノコギリ波の周波数に対して，サイン波の周波数を2倍にとってみましょう．周波数比は1:2です．図20がその波形です．

次に周波数比を1:5にしてみましょう(図21)．1:2のときにくらべて，ますます複雑な波形になります．

このように，ノコギリ波に同じ振幅のサイン波を加算した場合でも，サイン波の周波数を変えていくと記憶回路から読み出される波形も大きく変化します．このようにノコギリ波：サイン波の周波数比は，自由に変えられます．また，この比は整数比でなくてもよく，たとえば，1：8.265のような半端な数字でもかまいません．例として図22に，1:2.5, 1:3, 1:4の場合に読み出された波形を載せておきます．これらは，加算したサイン波の振幅を一定にして周波数比だけを変えたときの例です．

まとめとしてここで，特に重要なことを要約しておきましょう．

• ノコギリ波の周波数→読み出される波形の周波数を決定する．

• サイン波の振幅→読み出される波形のかたちを決定する．

• ノコギリ波とサイン波の周波数比→読み出される波形のかたちを決定する．

これで，FM音源の基本的な知識が身につきました．

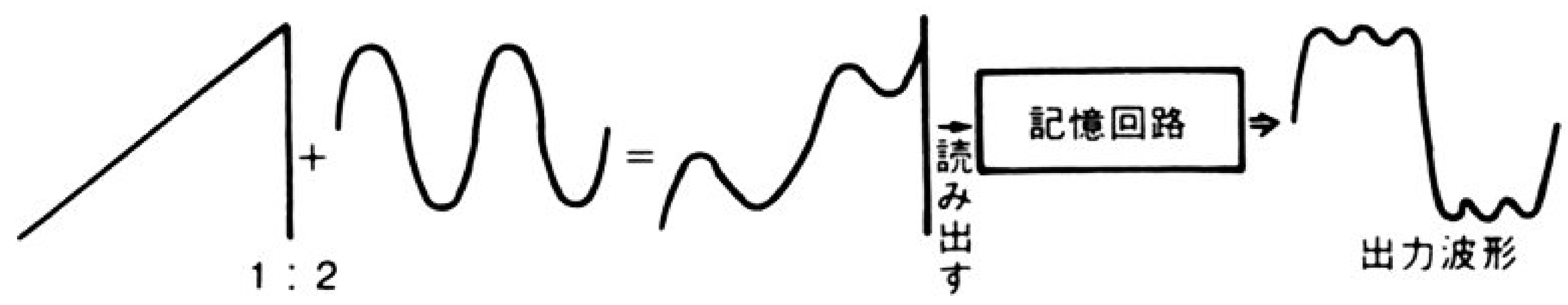


図20

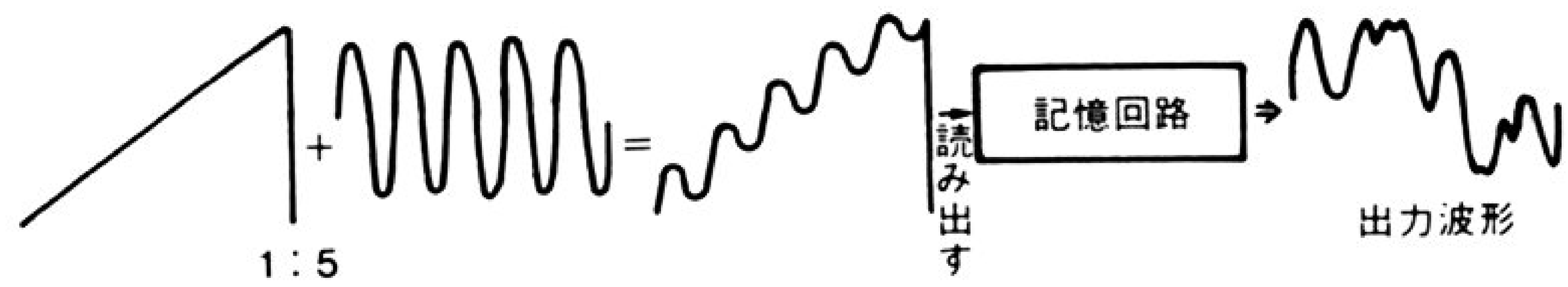


図21

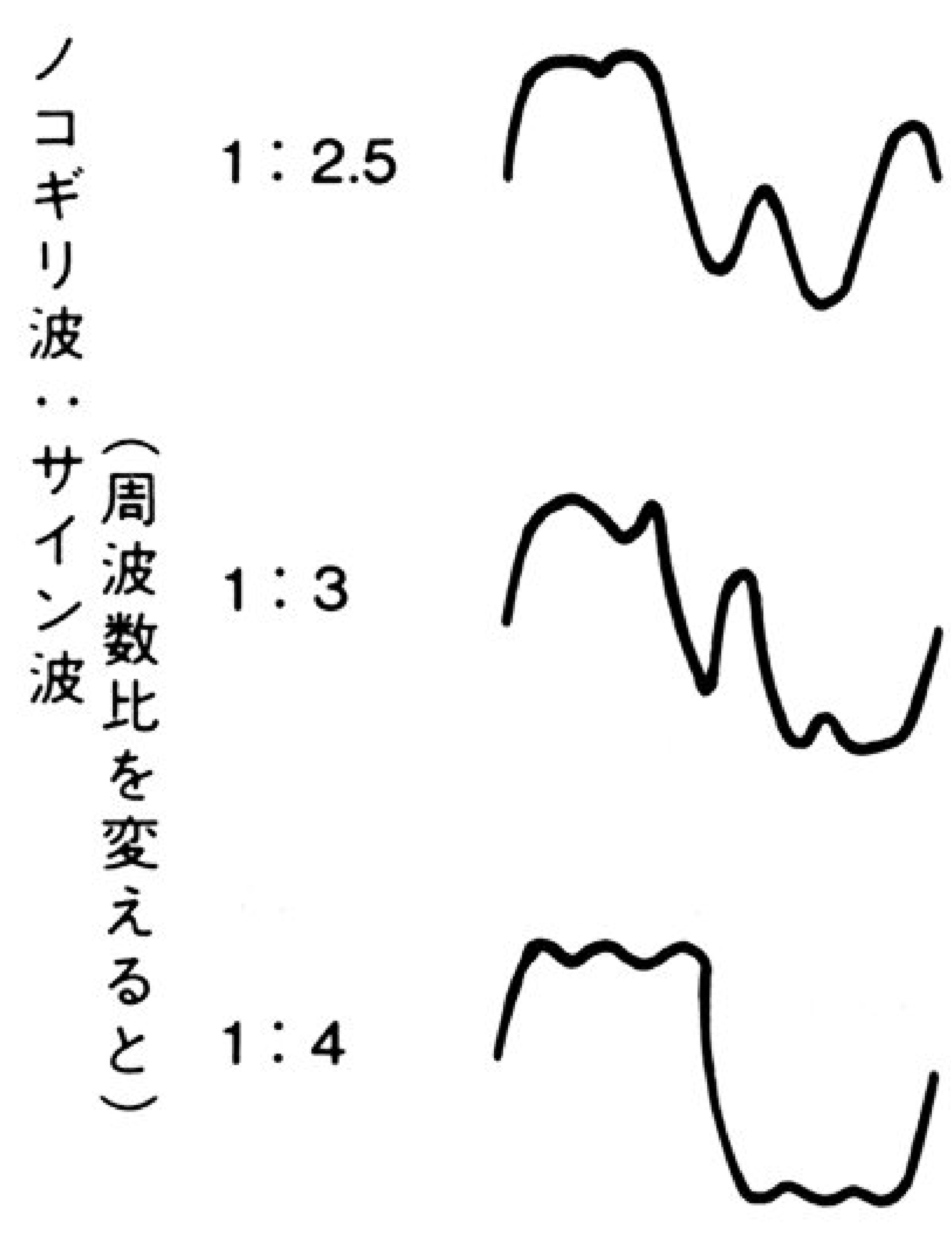


図22

8. オペレータの仕組み

FM音源の基本構成は，図23のように考えられます．周波数を決めるノコギリ波と，出力の波形を決めるサイン波など(どんな波形でもいい)が加算器で加算され，それで記憶回路を読み出します．記憶回路からの出力は，一たん掛算器を通って出力に出てきます．図24のように，振幅が一定の波形を掛算器に加えておいて，エンベロープ[注*]データ入力に出力される音の輪郭を入れると，波形の振幅はエンベロープデータに従います．

図23の基本構成を描きなおしてみると，図25のようになります． そして， これを"オペレータ"と呼んでいます．

加算器の２つの入力にはそれぞれ名前が付いていて，"ピッチ(＝周波数)データ入力"と"変調(＝波形)データ入力"になっています．

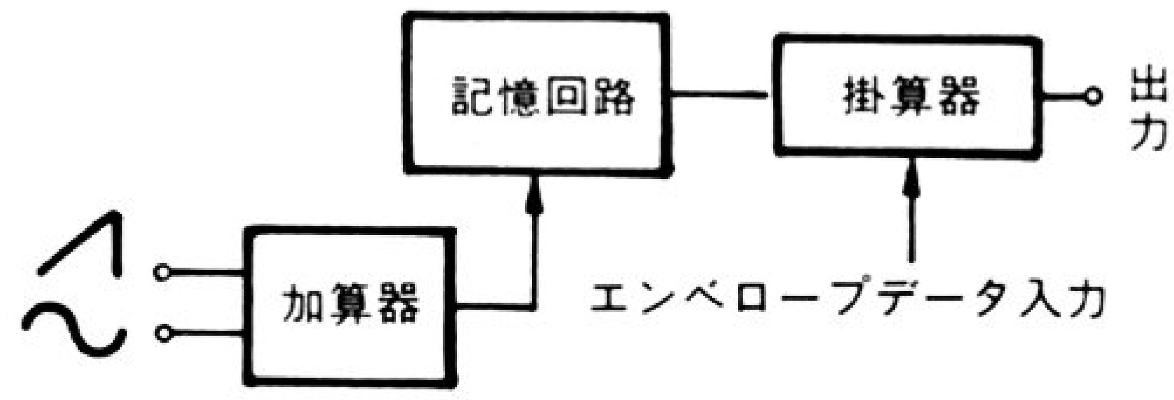


図23

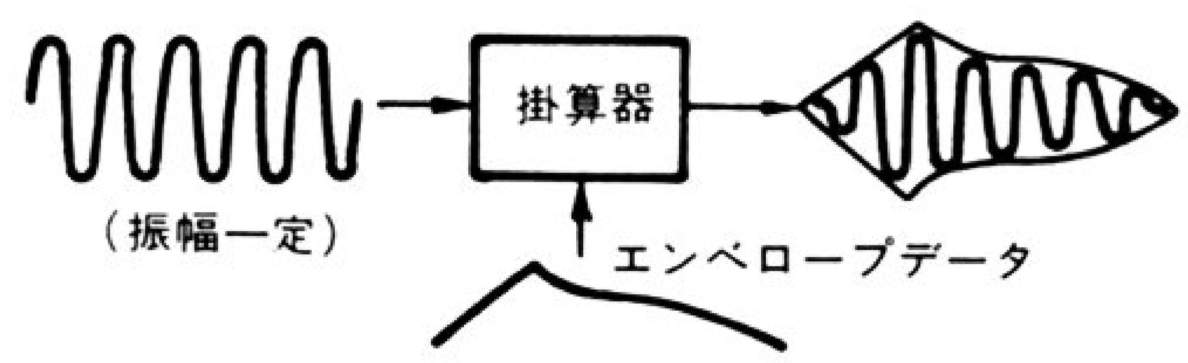


図24

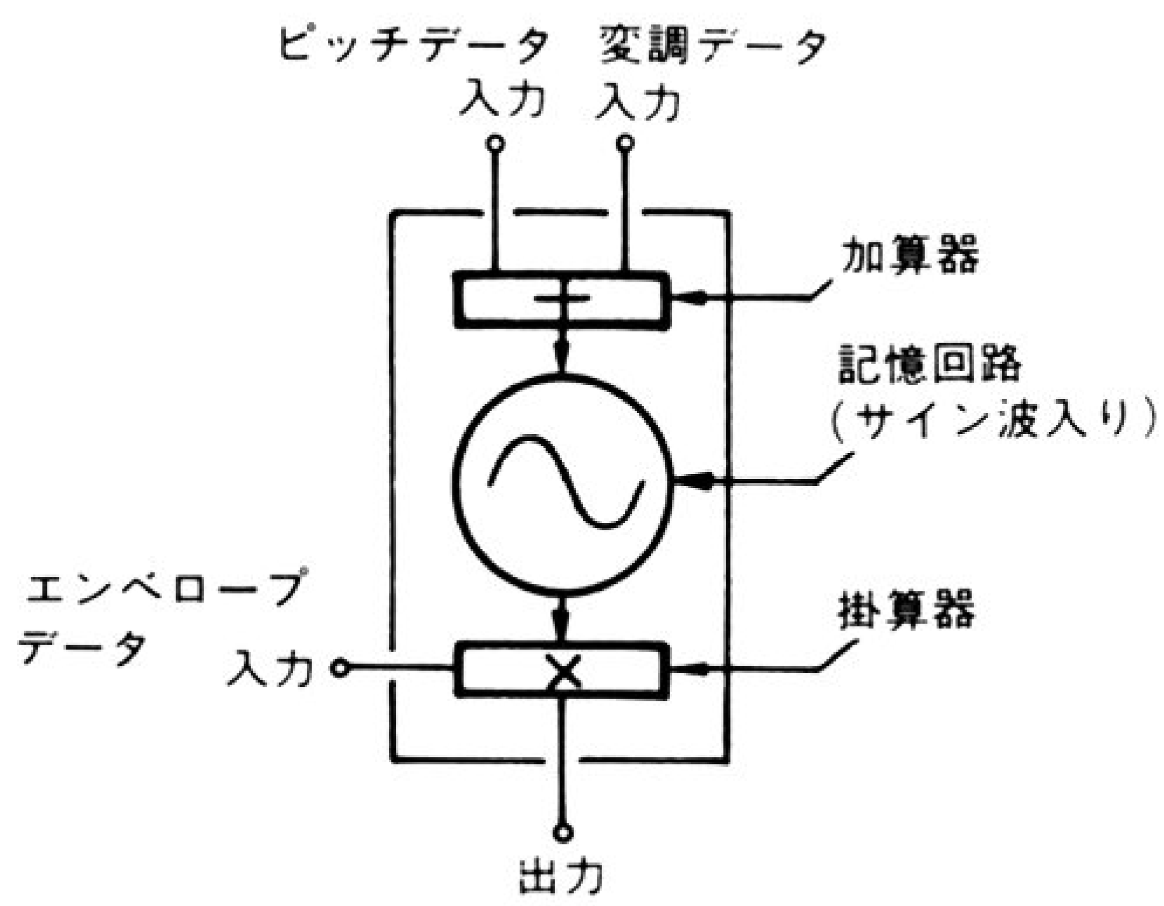


図25　オペレータ

注＊：エンベロープ

音の振幅は，発生した瞬間から消えるまで微妙に変化しています．その時間的な変化の様子をエンベロープといいます．エンベロープは，エンベロープデータを与えることによって，Key-onすると同時にエンベロープジェネレータ(EG)によって自動的に作られます．

9. オペレータの組み合わせ——FM音源での音の作り方

FM音源らしい音の作り方はどんなものなのでしょうか? 先ほど出てきた図16,17の動作をオペレータを使って描き表してみましょう．周波数は，ここでは一応440Hzとしておきます．

図26がその動作でオペレータOP.1は440Hzのノコギリ波で440Hzのサイン波を作り出しています．エンベロープデータ入力には何も加えていませんから，出力にエンベロープは付かず，一定振幅のサイン波が出っぱなしになります．

そのサイン波とモトになった440Hzのノコギリ波を，それぞれOP.2の変調データ入力とピッチデータ入力に加えるとOP.2の記憶回路から読み出された波形は図17と同じになります．このように実際のFM音

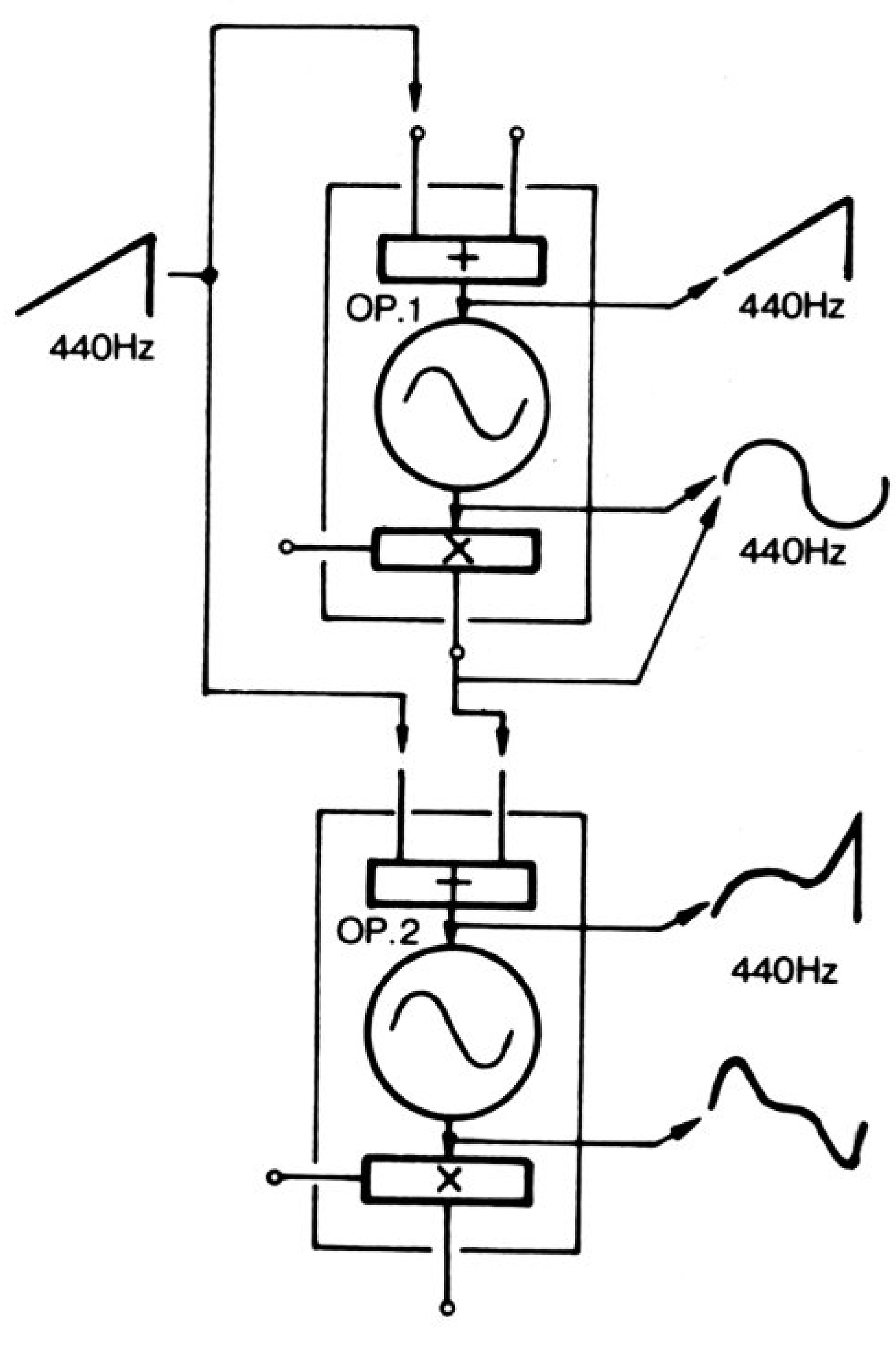


図26

源では，ノコギリ波だけを用意しておけば，どんな波形も作り出すことができます．

また図26で，OP．1のエンベロープデータ入力にある波形(の輪郭)を入れると，OP．1の出力のサイン波に輪郭が付き(図27)，これをOP．2に加えると，最終的に得られる波形(OP．2の出力)は時間とともに変化し，これは，音色の変化に当たります．つまり，音色に変調をかけたことになります．

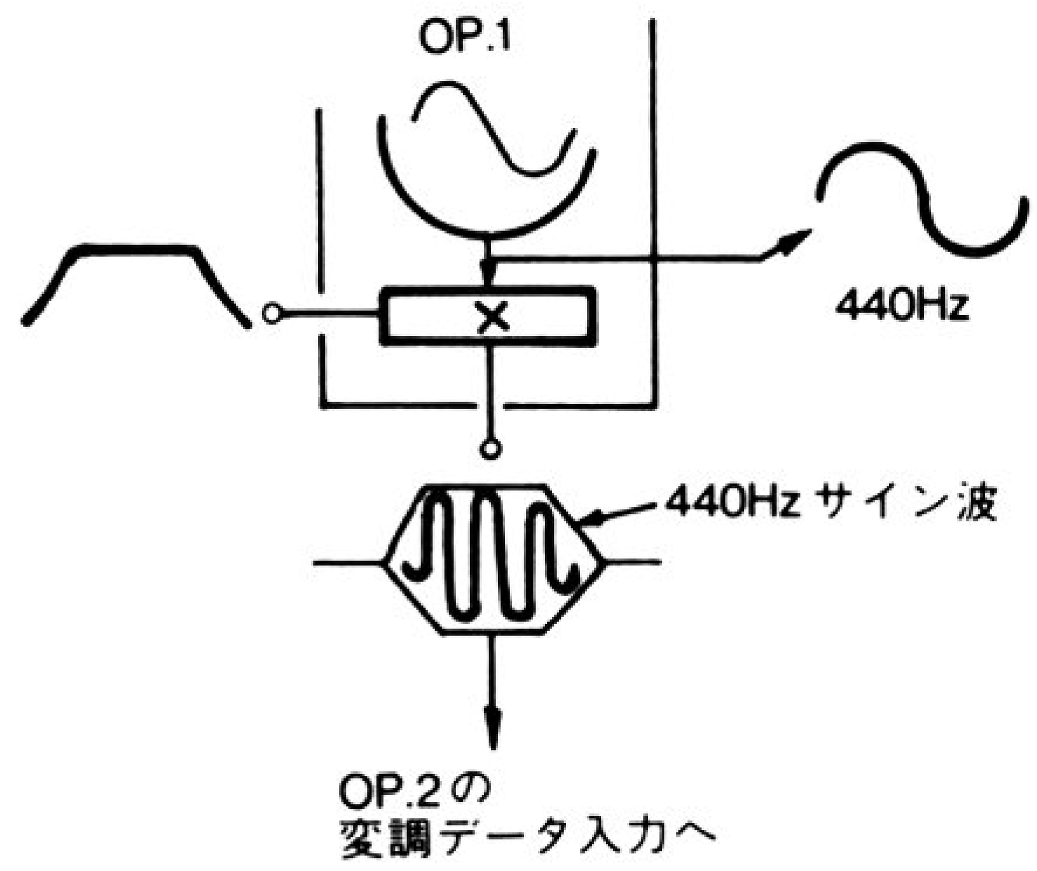


図27

これとは別に，エンベロープデータをOP．2に加えると，OP．2の出力にも輪郭が付き，これは音量の変化として聴こえます．

FM音源では，このようにオペレータを組み合わせることによっていろんな音色を作り出しています．

またFM音源ではオペレータを，その使われ方によって2つの種類に分けて考えるようにしています．図26の例ですと，OP．2は最終的な"音"になる波形を作り出していますが，オペレータがこのような使われ方をした場合，それを"キャリア"と呼びます．また，OP．1は，キャリアに与えるための変調用の波形(この場合はサイン波)を作っています．このようなオペレータを，"モジュレータ"と呼びます．

もう一つ，2個のキャリアで音色を作る例を挙げておきましょう．図28がそれで，FM音源ではいろんな周波数のノコギリ波を同時に発生させることができることを利用しています．先ほどの図26の例を"直列形"とでも呼ぶなら，こちらは"並列型"になります．

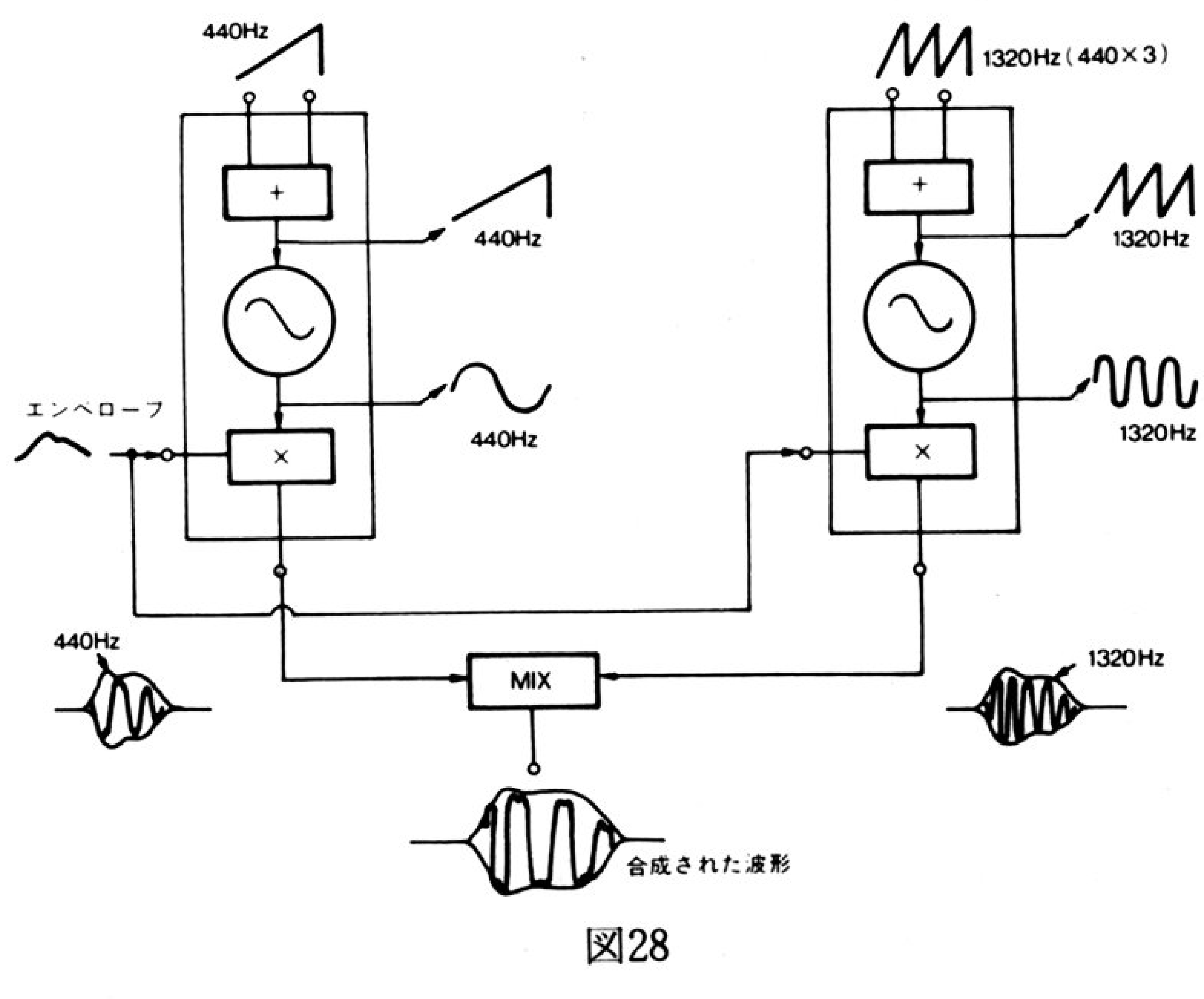


図28

このようなオペレータの接続の仕方をアルゴリズムといいます．そして，このアルゴリズムが音色を決定する大きな要素となっています．

PC-8801MKⅡMRに内蔵されているシンセサイザICは，1音に対して4つのオペレータを持ち，図29のように8つのアルゴリズムを選ぶことができます．

図29では，図25のオペレータを箱の形で表し，変調データ入力と出力以外は省略しています．複数の入出力があるものは，それぞれの入出力が加算されていることを意味します．たとえば図29の3:のアルゴリズムの場合，OP.2とOP.3の加算されたものがOP.4に入力されます．5:のアルゴリズムの場合，OP.2, OP.3, OP.4の加算されたものが出力されます．

周波数はピッチデータとしてオペレータに与えられます．本体に内蔵されているシンセサイザICでは，各々のオペレータに独立したピッチデータを与えていません．基本となる周波数があり，その周波数を何倍かしたものが各オペレータに与えられます．そして，その倍数をMultipleと呼びます．

たとえば，440HzのAの音をシンセサイザICに与えたときに，あるオペレータのMultipleが2だった場合，そのオペレータのピッチデータは440Hz×2=880Hzになります．Multipleの範囲は0〜15までの整数で，0を指定した場合，周波数は1/2になります．

（参考著書　デジタルキーボードハンドブック「日本楽器製造株式会社」）

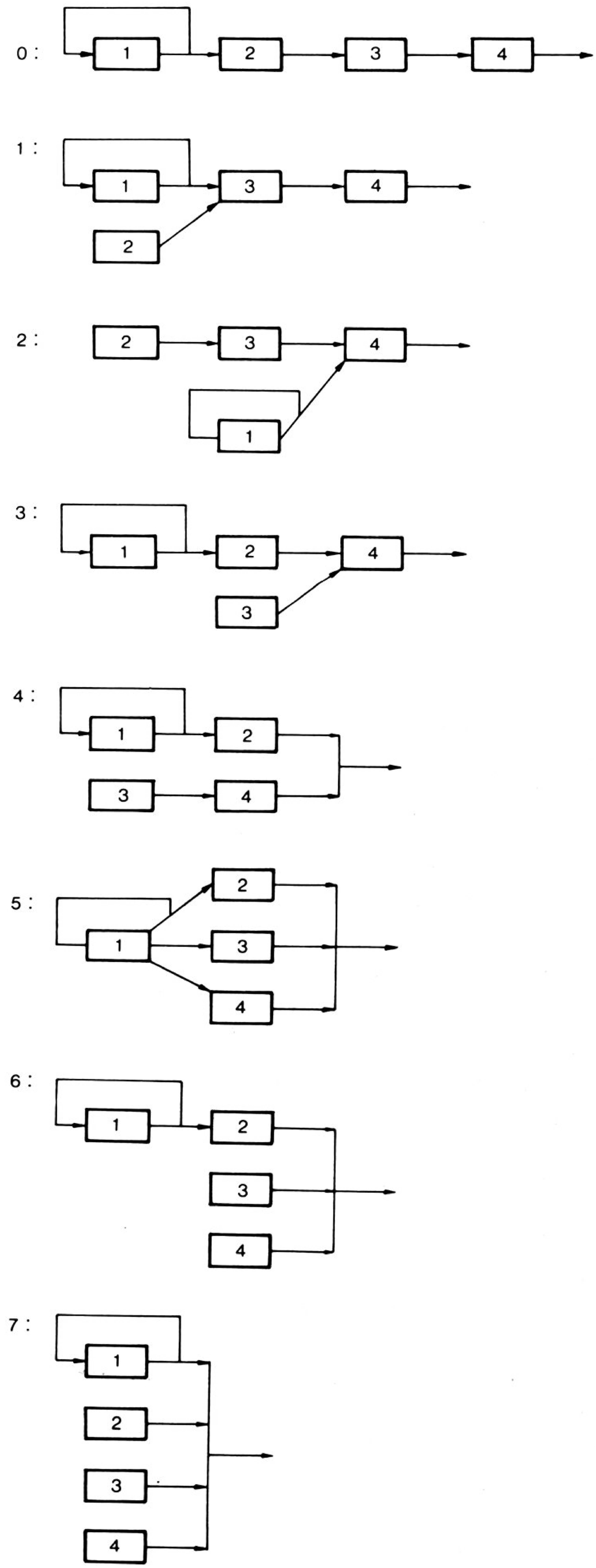


図29

資料3.2 音を作る

前節"FM音源"での知識をもとに，今度は実際に音を作ってみましょう．

音作りで最も重要なのは前節で述べた音の時間的変化，つまりエンベロープです．そして，作りたい楽器の特徴をよく理解することも大切なことです．

たとえばピアノであれば，鍵盤を押したとき鋭い音の立ち上がりがあり，その後続けて押していれば除々に音が消えていくエンベロープを持っています．また，倍音の構成も立ち上がりのときに多く，時間が経つにつれて倍音の数は少なくなり，一定の倍音構成に近づいていきます．以上のような特徴をつかんで音作りを進めていきます．

エンベロープを人工的に作る機能をEG(Envelop Generator)といいます．そのパラメータとしてAR(Attack Rate)，DR(Decay Rate)，SR(Sustain Rate)，RR(Release Rate)，SL(Sustain Level)などがあります．

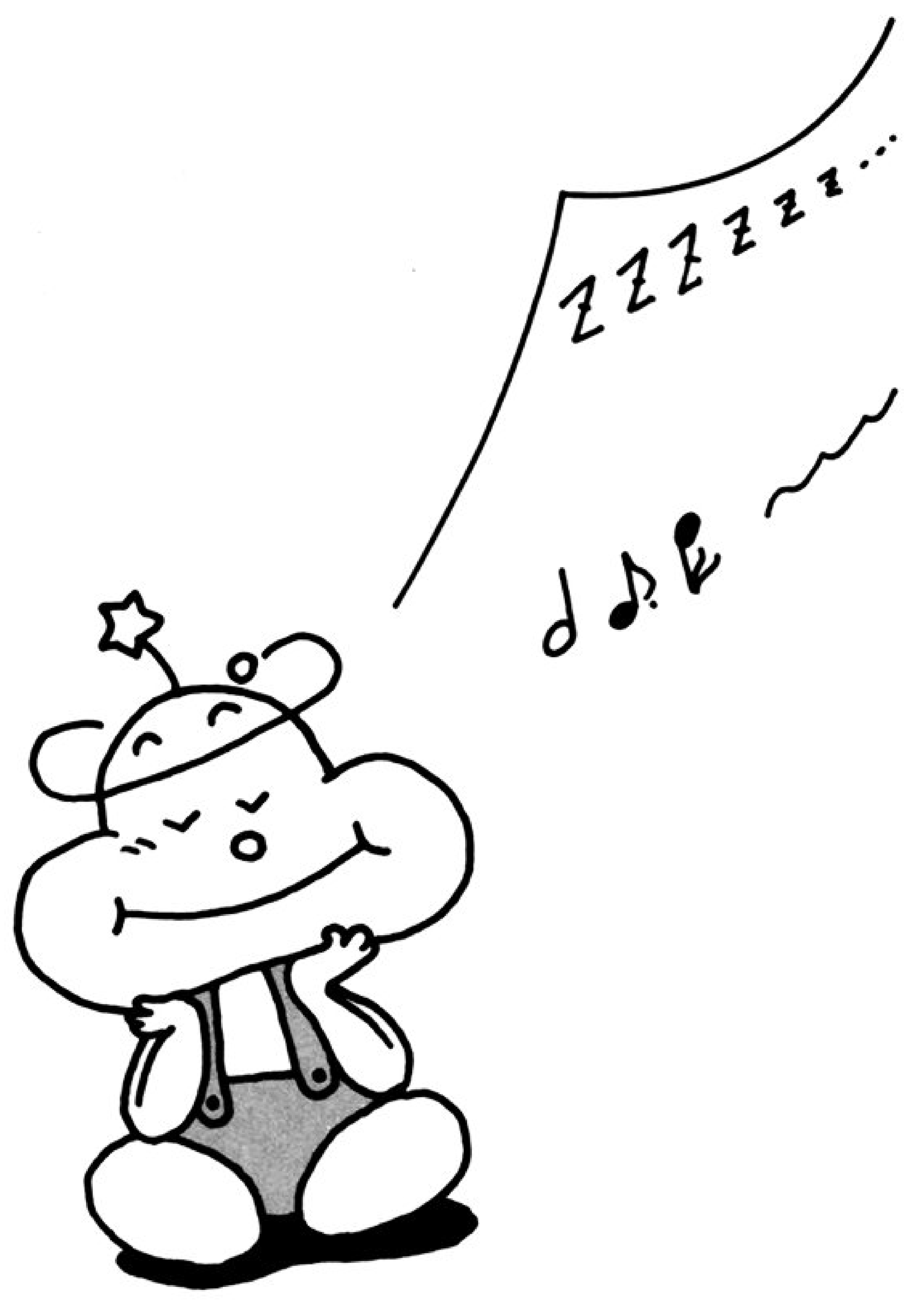
これらを設定していくためには配列変数を使います．この配列変数の各値は，それぞれ特別な意味を持っており，それらの値をCMD VOICE命令で音色として設定します．配列変数の意味は，BASICリファレンスマニュアル　4章のCMD VOICE文を参照してください．

まず配列変数を使うために，DIM文で宣言します．

```
DIM A%(4,9)⏎
```

FM音源で音を出すためには，オペレータというものが重要な役割りを果たしています．このオペレータは4つあり，それぞれをいろいろと組み合わせることによって出る音が全く違ってきます．その組み合わせをアルゴリズム(Algorithm)といい8つのパターンがあります(前ページのアルゴリズム表を参照してください)．

4つあるオペレータはモジュレータとキャリアに分かれ，これはアルゴリズムによってその対応が変わります．前の節で述



べたように，キャリアは出力する周波数がそのまま音の周波数になり，モジュレータは出力する周波数が変調信号としてキャリアに対して出力されます．

キャリアとモジュレータのパラメータの効果は，右のようになっています．

項目	関与するハラメータ	設定値による音の変化 値 MIN ↔ MAX
キャリアの出力レベル	各オペレータのOutput Level（AR, DR, SR, RR, SL）	音量大 ↔ 音量小（注：Levelは0が最大となります）
モジュレータの出力レベル		丸い音色↔明るい音色
モジュレータのフィードバックレベル	Feed back	普通の音色↔鋭い音色ノイズ
キャリアの周波数	各オペレータのDetune, Multiple	ピッチ低↔ピッチ高
モジュレータの周波数		近い倍音↔離れた倍音

次に，8つのアルゴリズムの特性を説明します．

Ⅰ）キャリア数1のアルゴリズム
　（番号0〜3）

1つのキャリアに対して残りのすべてのオペレータがモジュレータとなるわけですから，強力に明るい音になります．また，このアルゴリズムは極端な音色変化だけでなく，モジュレータの出力レベルを控え気味にすれば，複雑な波形を持つ微妙な音色を作ることも可能であり，ソロ楽器向きといえます．

Ⅱ）キャリア数2のアルゴリズム（番号4）

キャリア2つのアルゴリズムは，音作りの結果を想像しやすいわりに，幅広い音色が作れるオールマイティなパターンです．2つのキャリアのピッチをずらしてコーラス効果を出したり，一方で基本的な音色を作り，他方で味付け的な効果（たとえば，息の音）を出したりすることも可能です．

Ⅲ）キャリア数3, 4のアルゴリズム
　（番号5〜7）

厚みのある落ち着いた音色を作るのに向いています．すべてのキャリアのピッチを少しずつずらしてやると，多数の楽器を同時に弾いているようなコーラス効果が作れます．また，アルゴリズム7のように4つのオペレータの出力をすべて加えあわせるパターンは，オルガン系の音色を作るのに最適です．

それでは，これからエレクトリックオルガンの音を作ることにしましょう．

そこでまず初めに，これら4つのオペレータをONにして使用可能(音を出せる状態)にするために，次のようにしてください．

A%(0,1)=15⏎

この値の意味は，BASICリファレンスマニュアルのCMD VOICE文の配列変数の意味を参照してください．

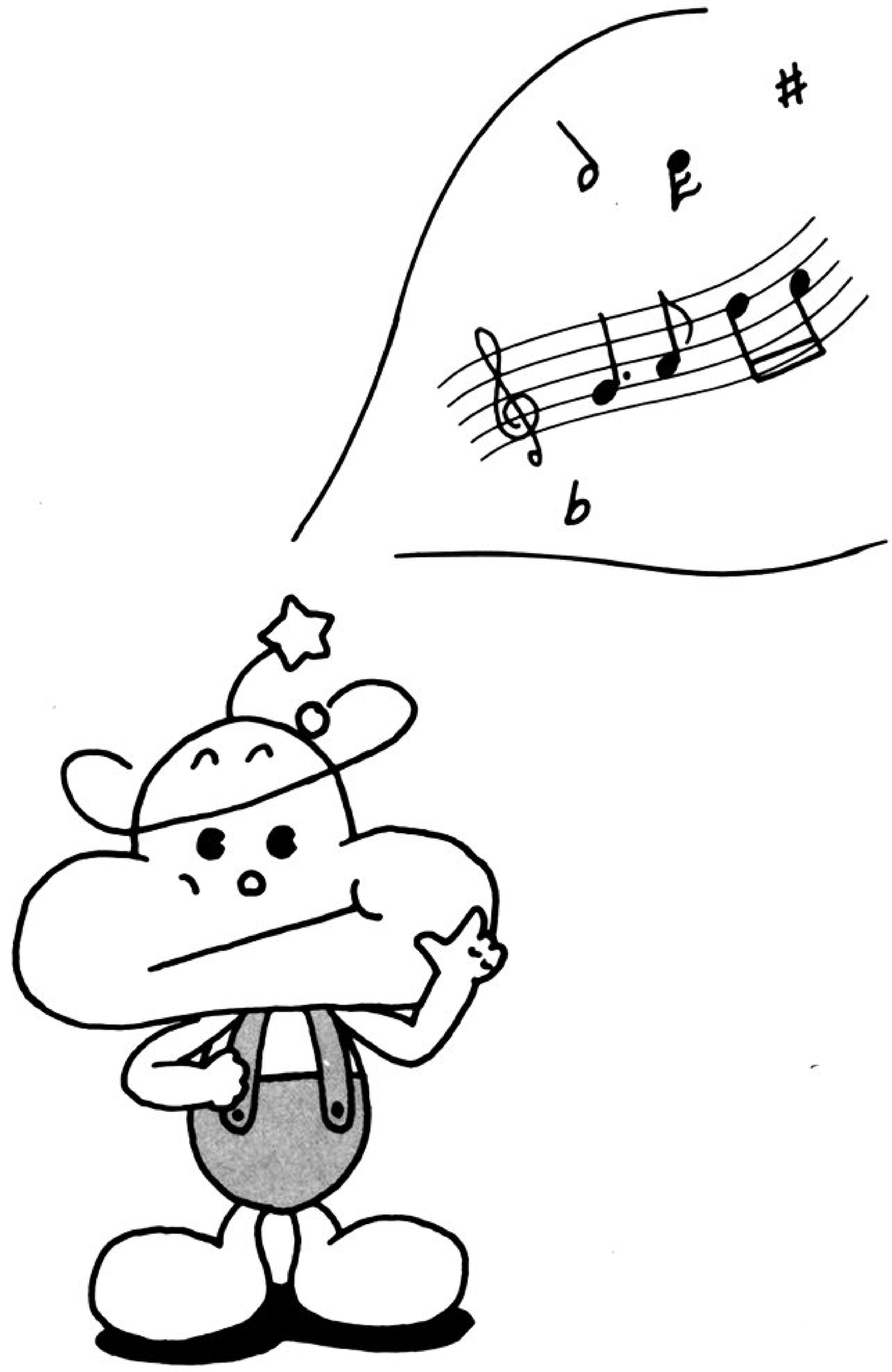
次にアルゴリズムを選ぶわけですが，前にも述べたように，オルガン系はアルゴリズム7が良いのですが，ノイズを付けてボリューム感のある音を作るためにアルゴリズム5を選びます．モジュレータであるオペレータ1がすべてのキャリア(オペレータ2, 3, 4)に対して変調をかけることによってノイズを付けることができます．

アルゴリズムの設定は配列(0, 0)で行い，フィードバック(Feed back……自分自身に対して変調をかける度合を設定する．)の値も一緒に設定します．このフィードバックはオペレータ1に働くもので，先ほども述べたように歪んだ感じを出してボリュームある音にするために，ここでは最大値の7にしておきます．この設定方法は，BASICリファレンスマニュアルのCMD VOICE文のFeed back/Algorithmを参照してください．

A%(0,0)=7*8+5⏎

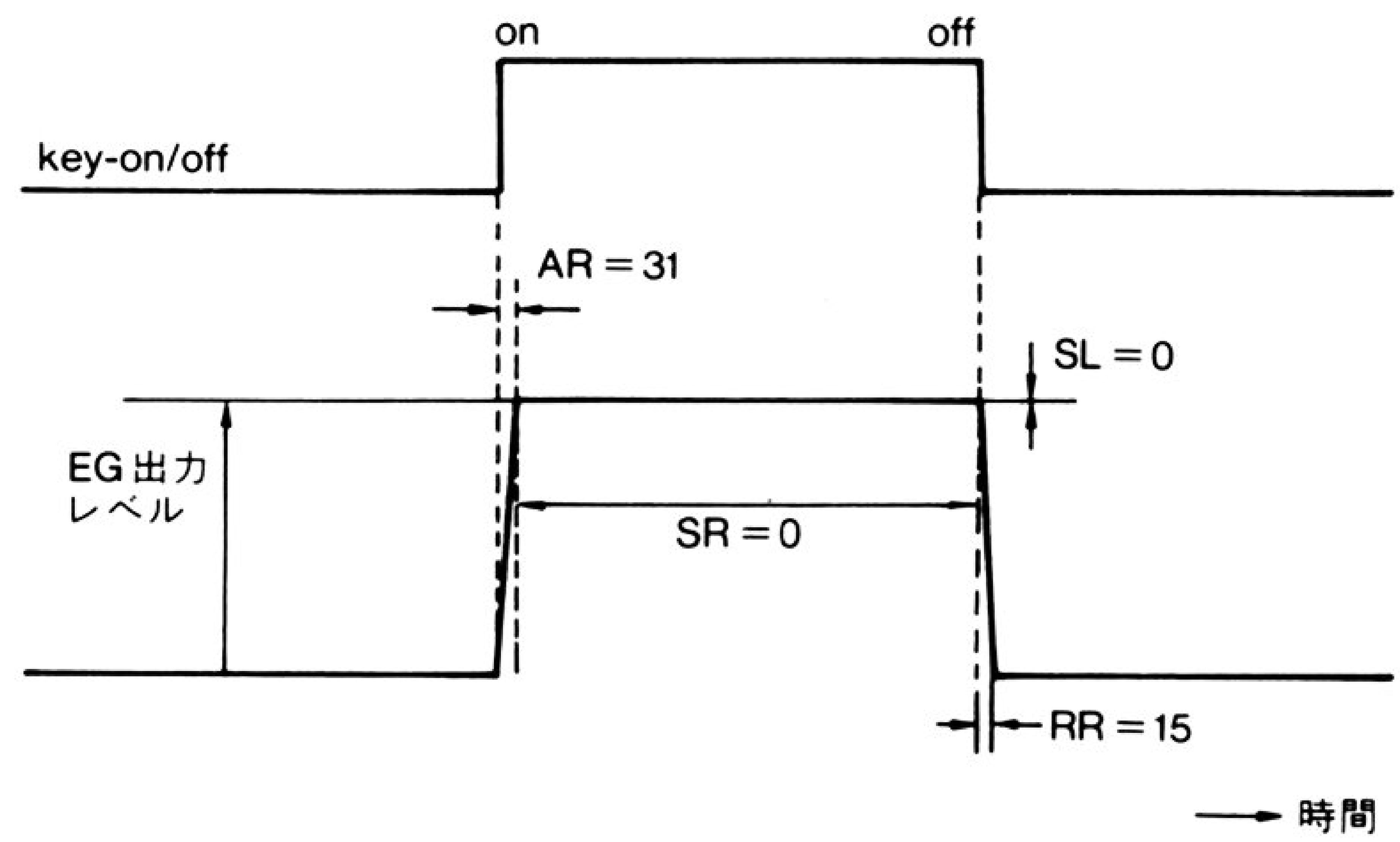
これでフィードバックとアルゴリズムの設定ができました．

次に音を出すためには，AR(Attack Rate)を設定します．ARを設定しない場合は，音の立ち上がる速さが設定されないので音が出ません．またRR(Release Rate)も設定します．このRRはKey-offしてからEG出力レベルが減衰していく速さを決めます．このRRを設定しなければ(RR=0のとき)，かなり長い間音が続きます．もしこのような状態になったときは，違う音色(RRが設定されている音色)を出して前の音を消してください．とりあえず両方とも最大値にしておきます．



```
FOR OP = 1 TO 4 : A%(OP, 0)
= 31 : NEXT OP ⏎
FOR OP = 1 TO 4 : A%(OP, 3)
= 15 : NEXT OP ⏎
```

ここでオペレータ1〜4の現在のエンベロープを概念的に表すと，右図のようになります．



音を出すために，CMD VOICE文でデータを音色として設定しましょう．

```
CMD VOICE A% ⏎
CMD PLAY "V15O4CDEFG" ⏎
```

どうですか．かなり激しい雑音がしたと思います．これはモジュレータのOutput Levelが高すぎるために起こります．モジュレータのOutput Levelが高すぎると，そのモジュレータがキャリアに対してかける変調度が大きくなりすぎて雑音のような音になります．

一方，キャリアの方のOutput Levelは他のキャリアとの兼ねあいもありますが，キャリアが1つだけの場合は，出力される音量の大小に関係します．

モジュレータのOutput Levelは作ろうとする音質にもよりますが，最大でも10くらいにとどめておいたほうがよいでしょう．

ここでは，モジュレータはオペレータ1だけでオペレータ1には，フィードバックを最大にかけているので，Output Levelを30に設定しておきます．

```
A%(1, 5) = 30 ⏎
```

もう一度音を出してみましょう．次のように入力してください．

```
CMD VOICE A% ⏎
CMD PLAY "V15CDEFG" ⏎
```

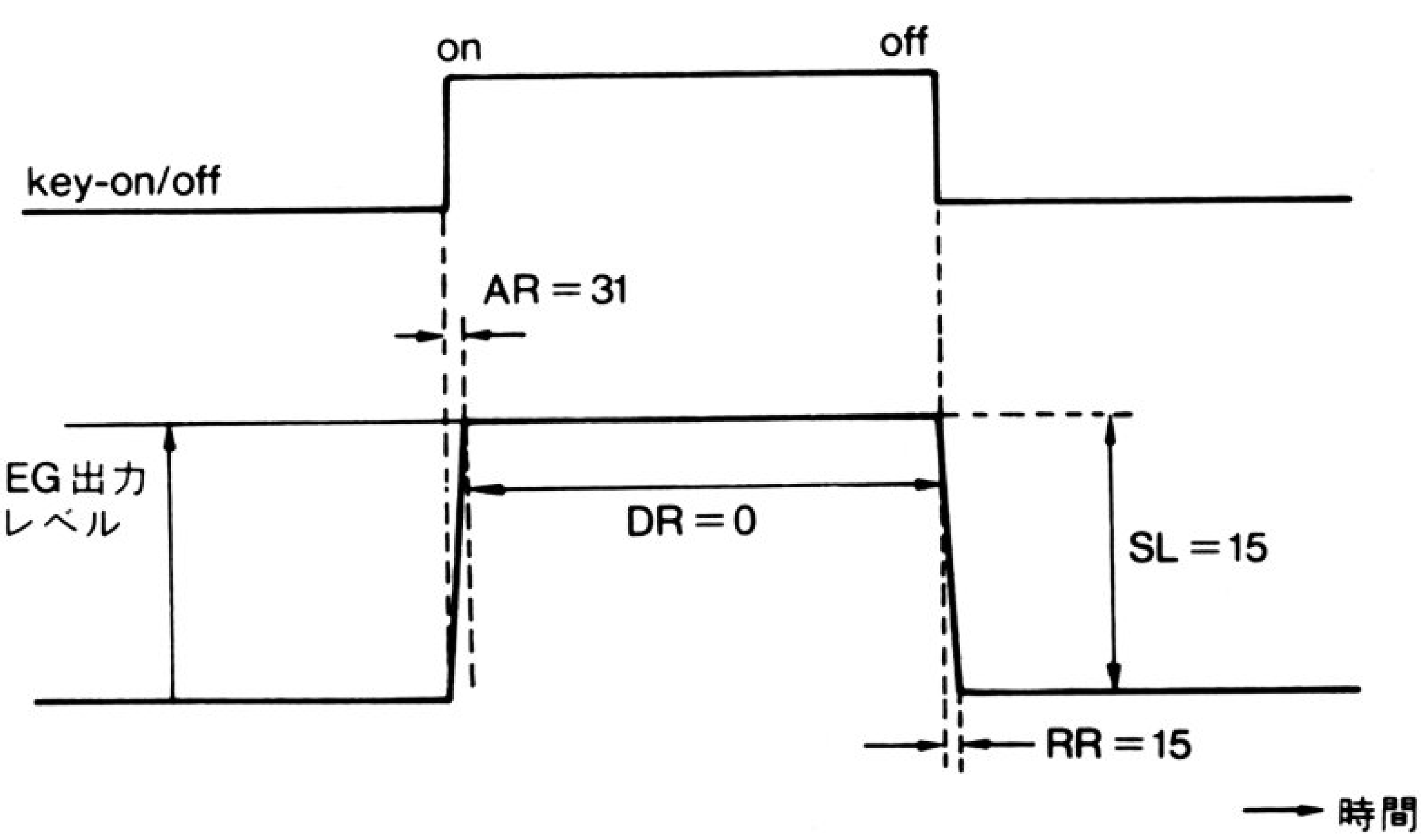
いくらかいい音になったと思います．まだ実際のエレクトリックオルガンの音とは全く違っていますが，細かい修正はあとにして，先に次のパラメータを設定することにします．

次はDR(Decay Rate)を設定します．Key-onされてARの値によりEG出力レベルが最高になってからこのレベルが下がっていく速さを，DRとSR(Sustain Rate)で決めます．このDRとSRはSL(Sustain Level)によって分かれます．EG出力レベルが最高からSLに達するまでの速さがDRで，SLから出力が0あるいはKey-offされるまでがSRの値になります．このSLを0(減衰量を0とします．したがって，出力レベルは最高値)にするとEG出力レベルが最高に達した時点からSRになりますので，DRの値は意味を持たなくなり，またそれとは逆にSLを15にすると，EG出力レベルが0になるまでDRの値により減衰していくのでSRの値は意味を持たなくなります．

各オペレータのSLを15にします．

```
FOR OP = 1 TO 4 : A%(OP, 4) =
15 : NEXT OP ↵
```

オペレータ1〜4の現在のエンベロープを概念的に表すと右図のようになります．



これでだいたいの準備ができました．それではこれからオペレータを分けて部分的に修正して音を作り上げていきましょう．

まず，オペレータ4だけONにします(このオペレータと設定値の関係はBASICリファレンスマニュアル　4章　CMD VOICE文を参照してください)．

```
A%(0, 1) = 8 ↵
```

現在，Multiple(周波数比)の値が0に設定されています．これを1にしておきます．このMultipleは，入力された周波数を何倍にするかを決めるものです．

```
A%(4, 7) = 1 ↵
```

これで音を出してみましょう．

```
CMD VOICE A% ↵
CMD PLAY "CDEFG" ↵
```

あまり音に表情が感じられませんね．これは正弦波によって音が作られているからです．

そこで，次のキャリアであるオペレータ3をONにしてハモった感じを出すためにMultipleを2にして音を出してみましょう．

```
A%(0,1)=4:A%(3,7)=2 ↵
CMD VOICE A% ↵
CMD PLAY "CDEFG" ↵
```

音程は高くなりましたが，やはりこれも正弦波1つなので単純な音ですね．今度はオペレータ3と4をONにして音を出します．

```
A%(0,1)=12 ↵
CMD VOICE A% ↵
CMD PLAY "CDEFG" ↵
```

今度は，原音のオペレータ4と倍音のオペレータ3の音がミックスされてオルガンらしい音色に近づいてきました．次に最後のキャリアであるオペレータ2で高い周波数を出すために，Multipleを10に設定します．

このとき，オペレータ2をONにするのを忘れてはいけません．

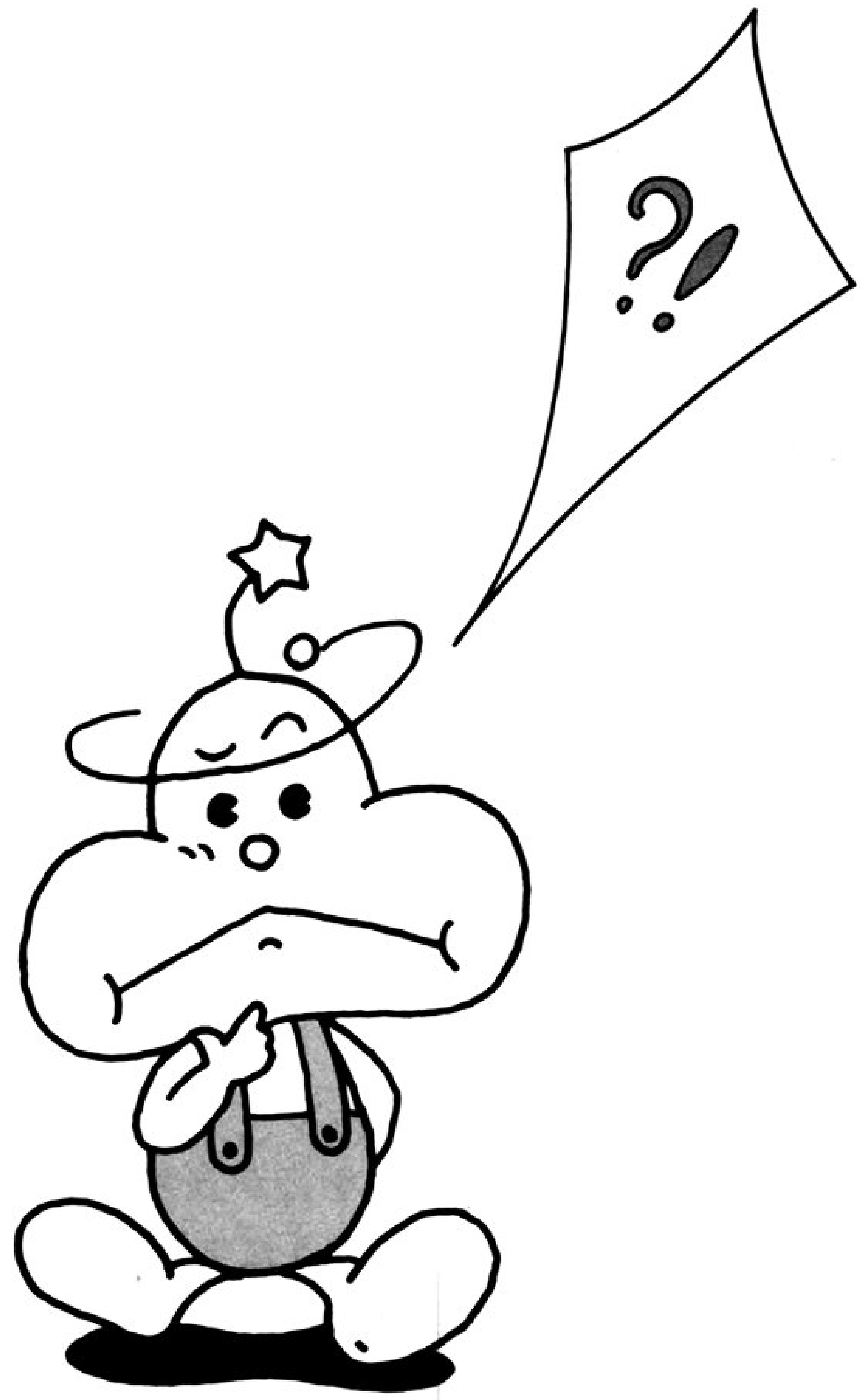
```
A%(0,1)=14:A%(2,7)=10 ↵
CMD VOICE A% ↵
CMD PLAY "CDEFG" ↵
```

高音部は出るようになりましたが，音程がくるっているように聞こえるのでもう少し下げます．

```
A%(2,7)=8 ↵
CMD VOICE A% ↵
CMD PLAY "CDEFG" ↵
```

これで良くなりましたね．次にモジュレータであるオペレータ1をONにして音を出してみます．

```
A%(0,1)=15 ↵
CMD VOICE A% ↵
CMD PLAY "CDEFG" ↵
```



多少いびつな音に聞こえますので，オペレータ１のMultipleを５にしてOutput Levelを少し下げます．

```
A%(1,7)=5:A%(1,5)=40⏎
CMD VOICE A%⏎
CMD PLAY "CDEFG"⏎
```

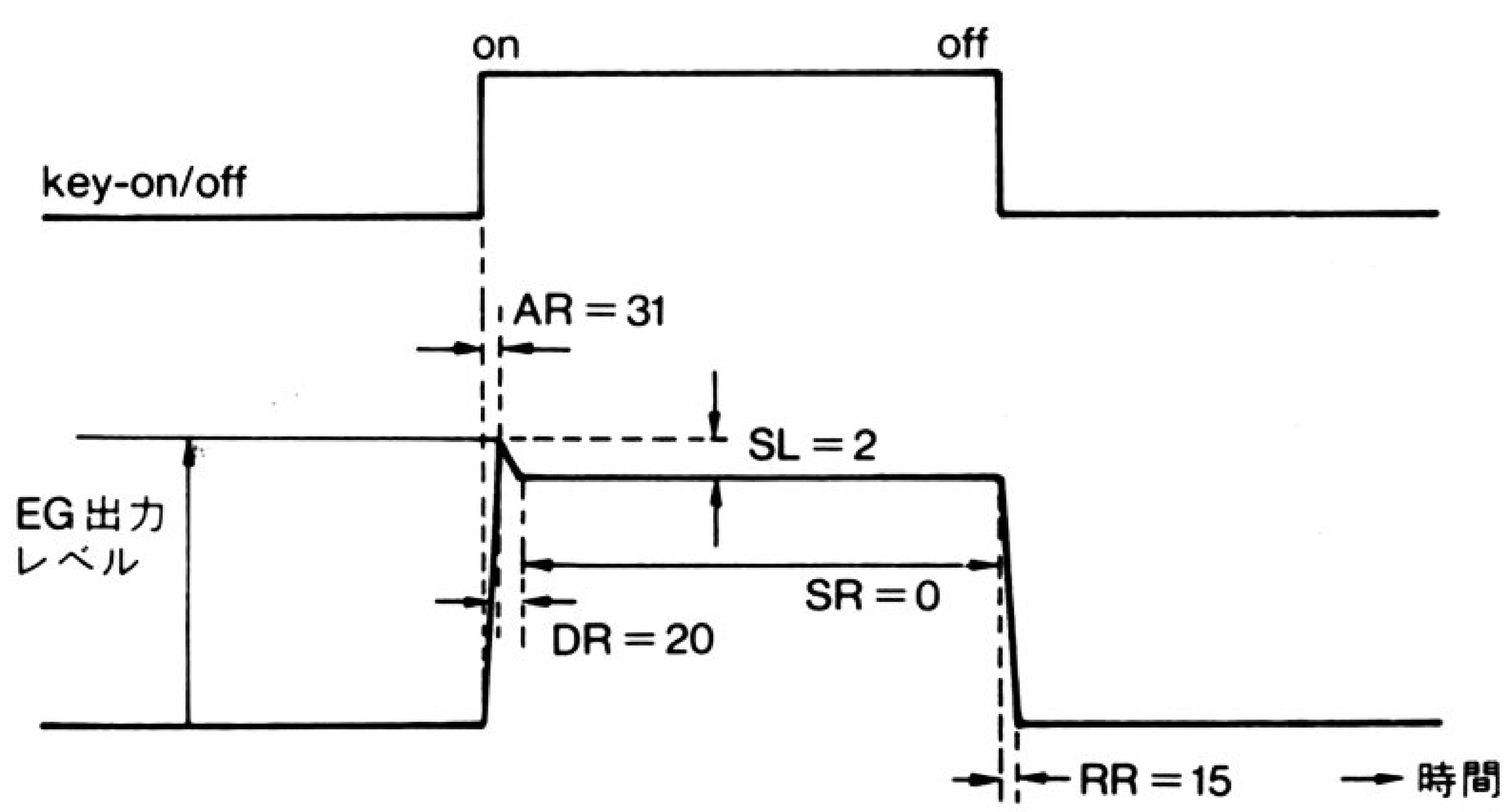
ずいぶんとオルガンらしい音になってきました．しかし，まだ歪んだ感じがしますのでオペレータ2, 3のOutput Levelを下げます．

```
A%(2,5)=20:A%(3,5)=30⏎
CMD VOICE A%⏎
CMD PLAY "CDEFG"⏎
```

まだ歪んだように聞こえますね．これはオペレータ１による変調がオペレータ2, 3, 4全部に同様にかかっているためです．そこでオペレータ１の出力レベルを音が立ち上がった状態から少し下げるようにします．それには，SLを２に設定すればよいでしょう．そしてDRが０ではEG出力レベルが下がらないので，立ち下がりを少し速めの20に設定します．

```
A%(1,4)=2:A%(1,1)=20⏎
CMD VOICE A%⏎
CMD PLAY "CDEFG"⏎
```

オペレータ１の現在のエンベロープを概念的に表すと右図のようになります．



これで少し丸みが出てきて，エレクトリックオルガンの感じになってきました．あとはもう少し音の幅を持たせるために，ディチューン(Detune)[注*]を設定して，周波数のピッチをずらします．

```
A%(1,8)=3:A%(2,8)=-1⏎
A%(3,8)=-3:A%(4,8)=3⏎
```

注＊：ディチューン(Detune)

FM音源のオペレータのピッチを微妙にずらすことによって，音色にうねりや深みを与えることができます．

オルガン，エレクトリックピアノ，ストリングスに，この機能を与えることによって，より自然な楽器の感じが得られます．

そのあとKeyboard Rate Scaling Depth注**を設定して，音程が高くなるほど各Rateによる変化速度が速くなるようにします．オペレータ2,3,4の各Rate値が最大か，あるいは0で設定されていなければKeyboard Rate Scaling Depthを設定しても意味がないので，ここではオペレータ1だけ3にします．

A%(1,6)=3⏎

そして最後に，LFOのパラメータを設定してビブラート(音程を細かく変化させる)やトレモロ(音量を細かく変化させる)などの効果を付けます．しかし，このLFOをあまり深くかけすぎると音程がくるうことがあるので注意してください．

このエレクトリックオルガンの音には，ビブラートだけでトレモロは付けませんが，トレモロをかけるためのパラメータの設定は，LFOスピードが1500〜2500くらい，Amsを3〜8，Amdを低くするとなめらかな効果が得られます．トレモロをかけるときは，最初にこの程度の値を設定しておいて，あとは音色によって修正していくとよいでしょう．

ここではビブラートをかけるために，ピッチ(Pmd, Pms)を変えます．細かいビブラートをかける場合，LFOスピードは3000〜4000くらいがいいでしょう．

エレクトリックオルガンは，それほど強くビブラートをかける必要はないので，Pmsを5，Pmdは10に設定します．

A%(0,4)=3000⏎
A%(0,7)=5⏎
A%(0,5)=10⏎

次にWave formですが，ビブラートには三角波が適していますので，2を設定して音を聞いてみましょう．LFOはやや高めで長い音にするとわかりやすくなります．

A%(0,2)=2⏎
CMD VOICE A%⏎
CMD PLAY "L1O6C"⏎

もう少しはっきり表すためにPmsを上げます．しかしこのままでは，かかりすぎるのでPmdを少し下げることにします．

注**：キーボードレイトスケーリングデプス(Keyboard Rate Scaling Depth)

音程によってエンベロープの各Rateを変化させます．自然の楽器のエンベロープは，ピアノの低音部と高音部を聞いてもわかるように音程によって微妙に異なっています．

このキーボードレイトスケーリングを設定することによって高音部での各Rateが速くなり，より自然な楽器の音に近い感じが得られます．

次のようにしてください．

```
A%(0,7)=6:A%(0,5)=8⏎
CMD VOICE A%⏎
CMD PLAY "L106C"⏎
```

これでほど良い効果が得られたと思いますが，もっと長い音を出してLFOの効果を確認しましょう．それには，"@W"を使うと便利です．

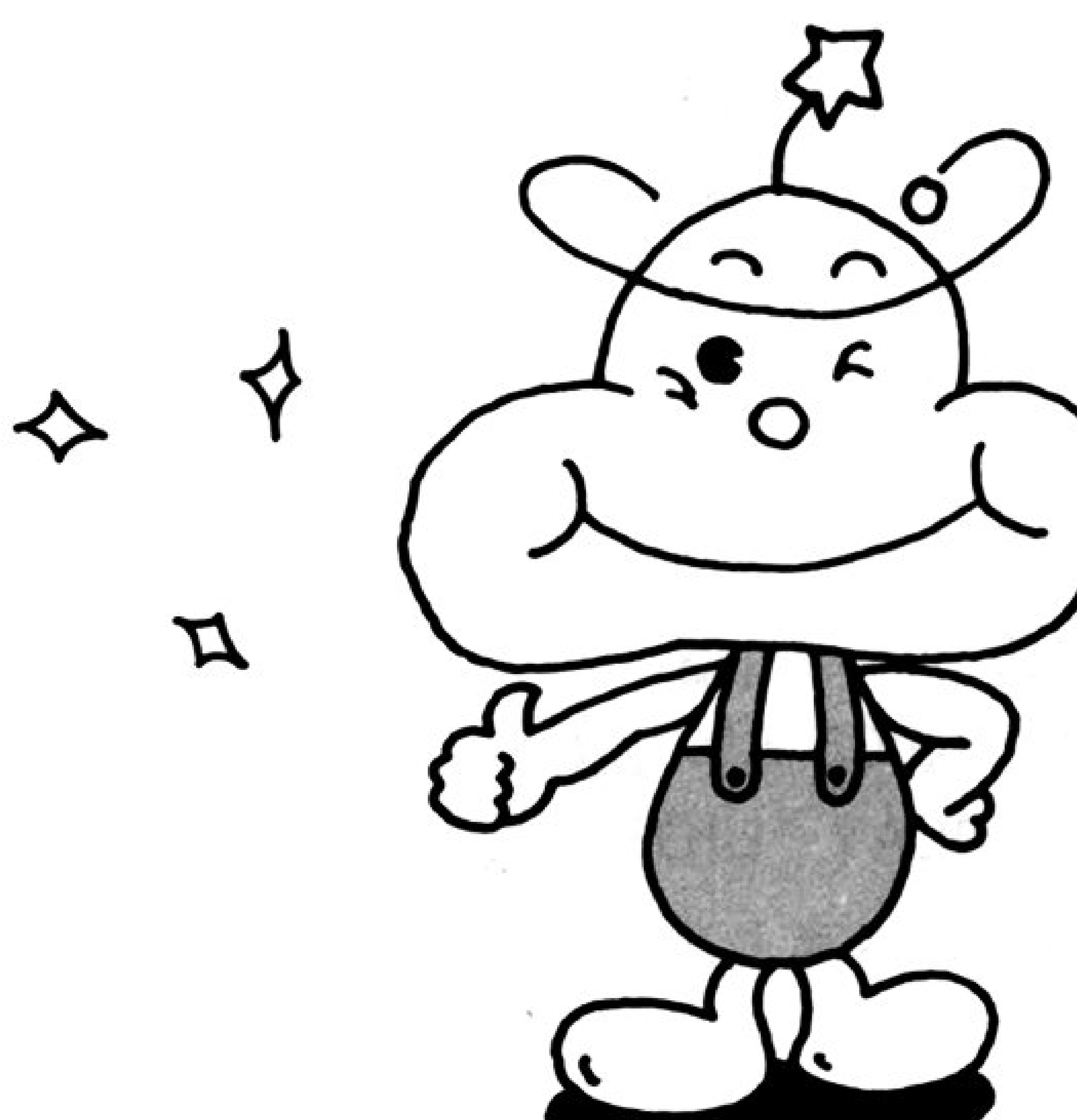
この"@W"は，指定されたときの状態を指定された長さだけ維持します．わかりやすく説明しますと，"@W"が指定されたときにKey-on（音が出ている）状態であれば，そのままKey-onされていて音が出ていますが，指定されたときにKey-offされていれば，そのままKey-offの状態，つまり音が出ない状態が続きます．よって，音程(C～B)の後に指定しても，この音程(C～B)は，Key-offしてから次のMMLパラメータを実行しますので，結局は"R"と同じ効果しか生みません．そこで長い音をずっと出す場合には，"Y"を使ってOPNレジスタに直接データを設定してKey-on状態を作り，その後に"@W"を指定します．OPNレジスタのKey-on, offを設定するレジスタ番号は40(&H28)ですので，そこへ240(&HF0)を設定するとKey-on状態になります(付録3　シンセサイザICの構造を参照してください)．

```
CMD PLAY  "Y40,240@W1@W1@W1
@W1"⏎
```

以上でエレクトリックオルガン音のできあがりです．

今回あげた例は，全く何もない状態から音色を作っていきましたが，作ろうとする音に近い音，あるいは特徴の似ているものを既存の音色から見つけ出して，その音をもとにして修正を重ね音を作るようにすれば，もう少し簡単に音を作ることができます．

その方法は，似ている音色をCMD VOICE COPY文で配列変数にパラメータをコピーして，その後は各パラメータの値と音とを比較しながら，今まで説明してきたように修正してください．



最後に，N_{88}-BASICシステムディスクの中には効果音や動物の鳴き声を出すプログラムが入っています．拡張子が"voi"となっているファイルを次のようにロードして，実行してください．

RUN "〈ファイル名〉. voi" ⏎

ただし，これらのプログラムはN_{88}-BASIC V2が起動した直後の状態で動作するように作られていますので，タートルグラフィック拡張命令は追加できません．

資料4. コントロールコード

• キー入力時の動作

16進	10進	キー入力	動　作
01H	1	[CTRL] + [A]	[HELP] と同じ
02H	2	[CTRL] + [B]	1つ前のワードに戻る
03H	3	[CTRL] + [C]	実行の中断([STOP] と同じ)
04H	4	[CTRL] + [D]	カーソル位置から1ワードを削除
05H	5	[CTRL] + [E]	カーソルの位置から，その行の終わりまでを消却
06H	6	[CTRL] + [F]	1つ先のワードへ進む
07H	7	[CTRL] + [G]	ブザーを鳴らす
08H	8	[CTRL] + [H]	1文字を削除([INS DEL] と同じ)
09H	9	[CTRL] + [I]	タブ位置までカーソルを移動.(タブ位置は8桁に設定されている.)([TAB] と同じ)
0AH	10	[CTRL] + [J]	ラインフィード，インサートモードで2行に分割
0BH	11	[CTRL] + [K]	カーソルをホームポジション（左上スミ)に移動([SHIFT] + [HOME CLR] と同じ)
0CH	12	[CTRL] + [L]	テキスト画面を消去する.([HOME CLR] と同じ)
0DH	13	[CTRL] + [M]	キャリッジリターン([↵] と同じ)
0FH	15	[CTRL] + [O]	プログラム実行中に画面の表示を無効にする
12H	18	[CTRL] + [R]	インサートモードにする.([SHIFT] + [INS DEL] と同じ)
13H	19	[CTRL] + [S]	実行を一時停止する
15H	21	[CTRL] + [U]	1行キャンセル
18H	24	[CTRL] + [X]	カーソルを行の最後に移す
1CH	28	[→]	カーソルを右へ移動
1DH	29	[←]	カーソルを左へ移動
1EH	30	[↑]	カーソルを上へ移動
1FH	31	[↓]	カーソルを下へ移動

• 画面表示のときの動作

PRINT 文で CHR$ 関数を用いて出力します.

例

```
print chr$(7);
```

16進	10進	動　　作
07H	7	ブザーを鳴らします.
09H	9	タブ設定までカーソルを移動させます. タブ位置は8桁ごとに設定されています.
0AH	10	カーソルを1つ下の行へ移動させます.
0BH	11	カーソルをホームポジション(左上スミ)に移動させます.
0CH	12	テキスト画面を消去します.
0DH	13	カーソルを行の左端へ移動させます.
16H	22	半角文字入力モードにします.*
17H	23	全角文字入力モードにします.*
1CH	28	カーソルを1つ右へ移動させます.
1DH	29	カーソルを1つ左へ移動させます.
1EH	30	カーソルを上の行へ移動させます.
1FH	31	カーソルを下の行へ移動させます.

* N_{88}-日本語BASICでのみ有効です.

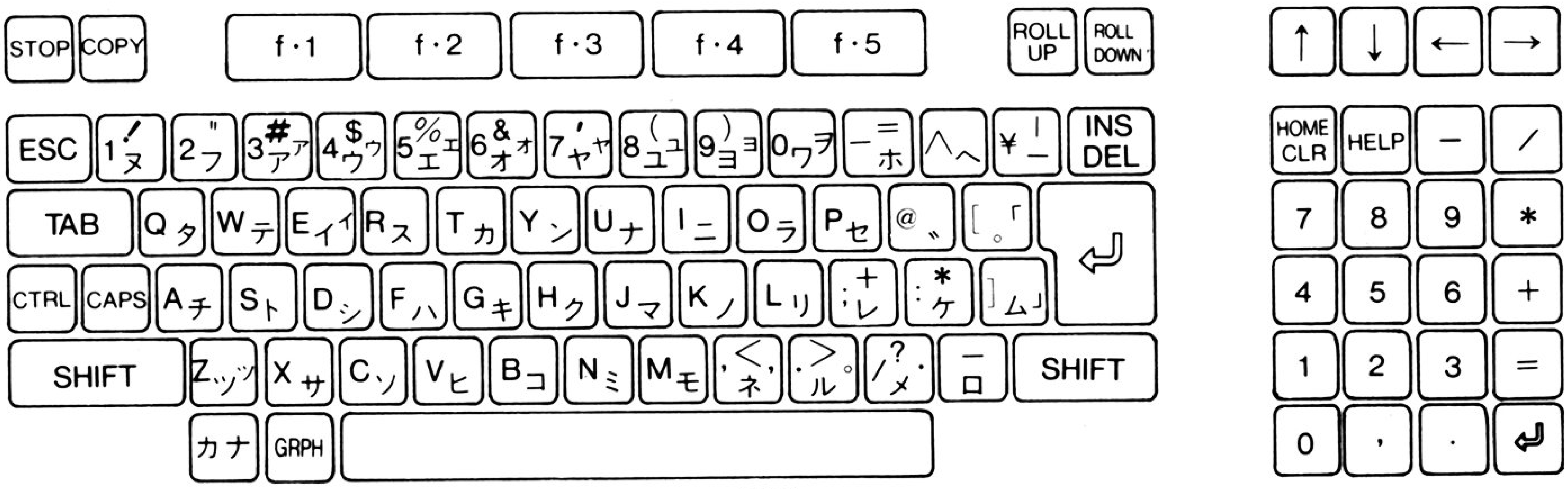
資料5. キャラクタコード

上位4ビット →

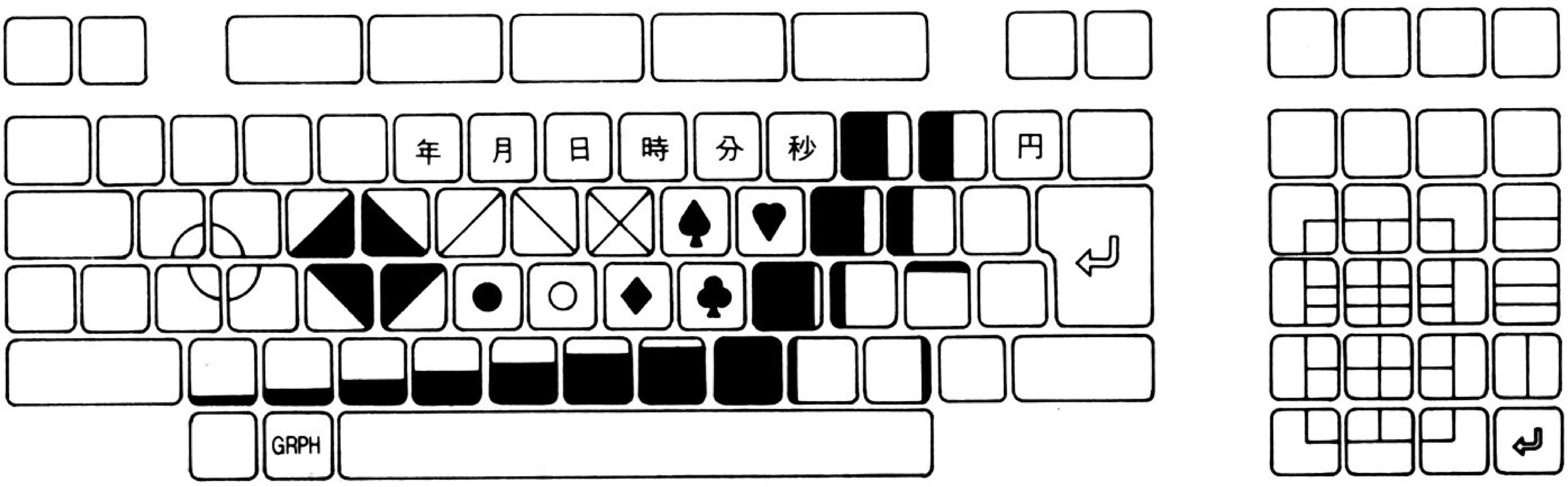
下位4ビット ↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE		0	@	P		p				ー	タ	ミ		
1	SH	D1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	SX	D2	"	2	B	R	b	r			「	イ	ツ	メ		年
3	EX	D3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	ET	D4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	EQ	NK	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	AK	SN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	BL	EB	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	BS	CN	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	
9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	
A	LF	SB	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	HM	EC	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	
C	CL	→	,	<	L	¥	l	¦			ャ	シ	フ	ワ	●	
D	CR	←	−	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	SO	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
F	SI	↓	/	?	O	_	o				ッ	ソ	マ	゜		

資料6. キーボードレイアウト



キーの配列



グラフィックシンボルキーの配列

資料7．ローマ字→かな変換規則

あ a	い i	う u	え e	お o	ぁ xa	ぃ xi	ぅ xu	ぇ xe	ぉ xo
か ka	き ki	く ku	け ke	こ ko	が ga	ぎ gi	ぐ gu	げ ge	ご go
さ sa	し si shi	す su	せ se	そ so	ざ za	じ zi ji	ず zu	ぜ ze	ぞ zo
た ta	ち ti chi	つ tu tsu	て te	と to	だ da	ぢ di	づ du	で de	ど do
な na	に ni	ぬ nu	ね ne	の no					
は ha	ひ hi	ふ hu fu	へ he	ほ ho	ば ba ぱ pa	び bi ぴ pi	ぶ bu ぷ pu	べ be ぺ pe	ぼ bo ぽ po
ま ma	み mi	む mu	め me	も mo					
や ya		ゆ yu		よ yo	ゃ xya		ゅ xyu		ょ xyo
ら ra la	り ri li	る ru lu	れ re le	ろ ro lo	っ かなの小文字「っ」は子音を重ねて変換します。 〈例〉・きっさ→kissa ・ちっそ→tisso ・りっとう→rittou				
わ wa				を wo					
ん nn wn n（次の子音）	●「ん」のあとにナ行またはヤ行のかながくるときは，「ん」は「nn」か「wn」で変換します。 〈正誤例〉・あんない　正annnai　誤annai ・いんよう　正innyou　誤inyou								

きゃ		きゅ		きょ	ぎゃ		ぎゅ		ぎょ
kya		kyu		kyo	gya		gyu		gyo
しゃ		しゅ	しぇ	しょ	じゃ		じゅ	じぇ	じょ
sha		shu	she	sho	ja		ju	je	jo
sha		syu	sye	syo	zya		zyu	zye	zyo
ちゃ		ちゅ	ちぇ	ちょ	ぢゃ		ぢゅ		ぢょ
cha		chu	che	cho	dya		dyu		dyo
tya		tyu	tye	tyo					
にゃ		にゅ		にょ					
nya		nyu		nyo					
ひゃ		ひゅ		ひょ	びゃ		びゅ		びょ
hya		hyu		hyo	bya		byu		byo
					ぴゃ		ぴゅ		ぴょ
					pya		pyu		pyo
ふぁ	ふぃ		ふぇ	ふぉ	ヴァ	ヴィ	ヴ	ヴェ	ヴォ
fa	fi		fe	fo	va	vi	vu	ve	vo
ふゃ		ふゅ		ふょ	●ヴァ行はカタカナとして入力され，これらをひらがなに変換することはできません。				
fya		fyu		fyo					
みゃ		みゅ		みょ	● 文字以外の変換 ー(長音) ← ー(マイナス記号) 。(句点) ← .(ピリオド) 、(読点) ← ,(カンマ) テンキーから入力した場合は，マイナス，ピリオド，カンマのまま入力されます。				
mya		myu		myo					
りゃ		りゅ		りょ					
rya		ryu		ryo					
lya		lyu		lyo					

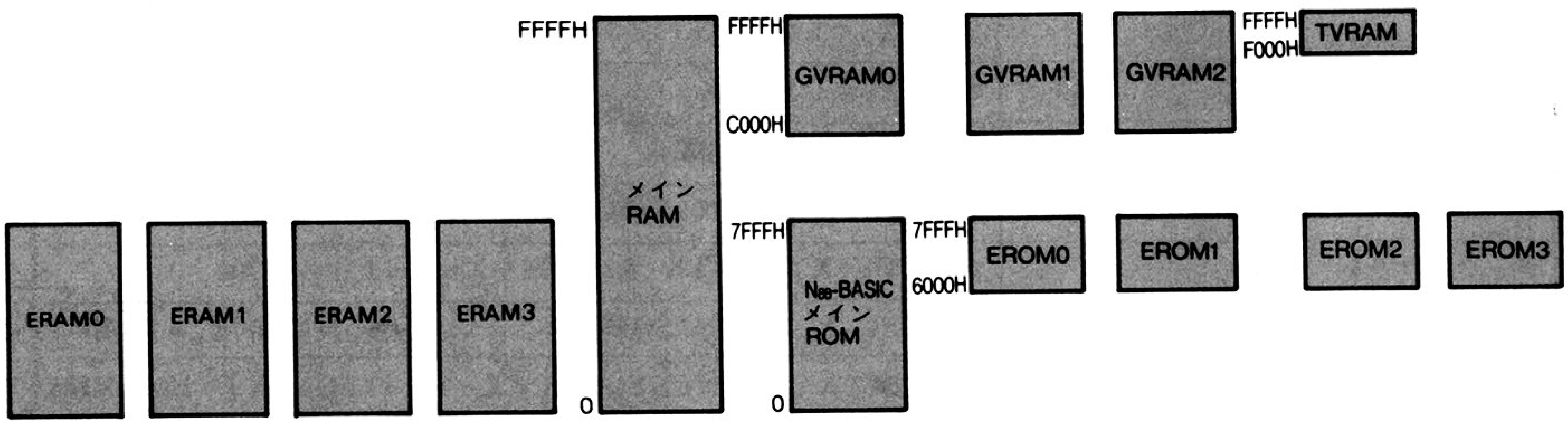
資料8. 誘導関数

BASICが"組み込み"関数として用意していない三角関数のうち，いくつかは，用意された関数を使って作り出すことができます．以下の数式を参考にしてください(誤差の範囲に注意が必要です)．

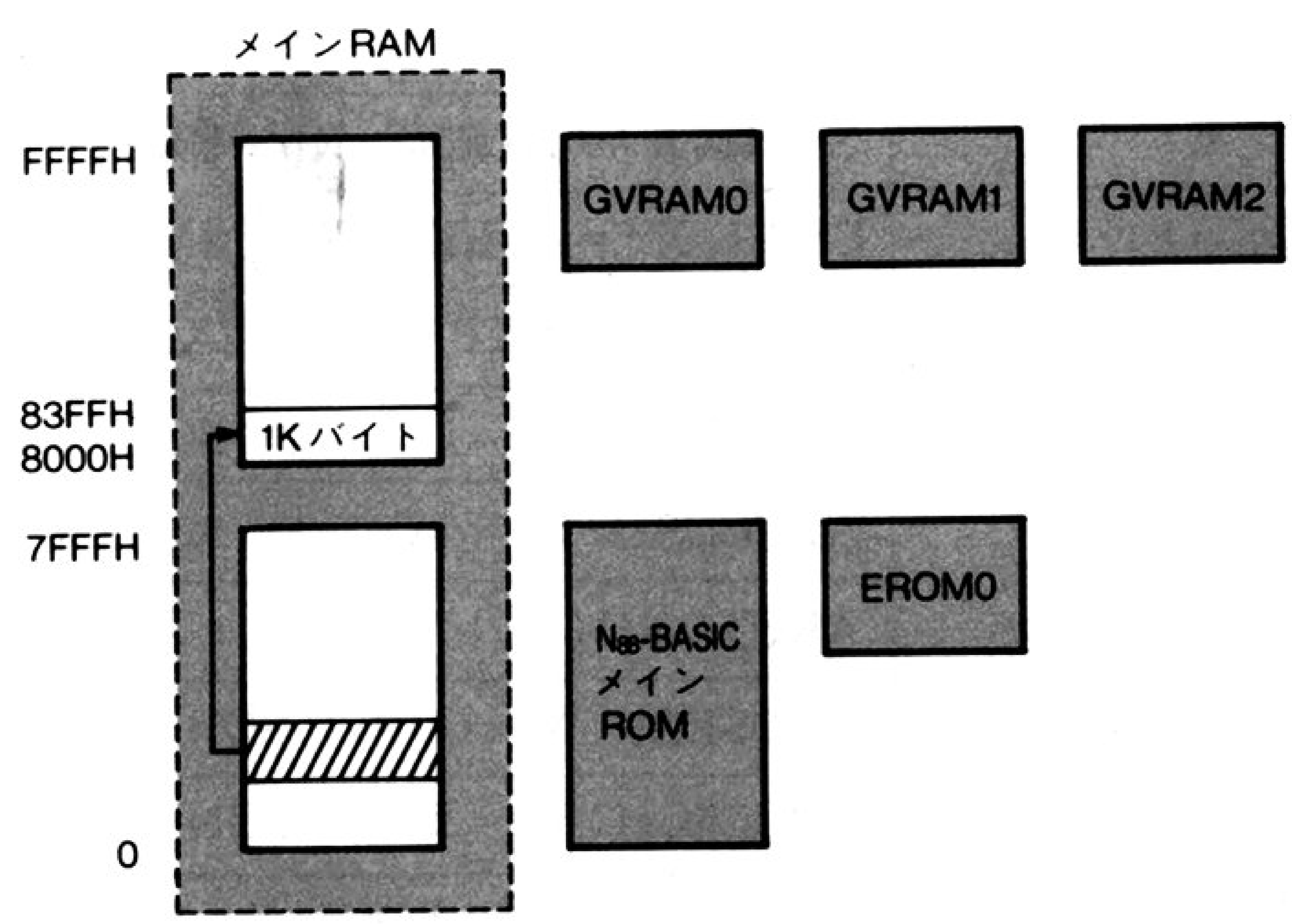
目的とする関数	組み込み関数からの誘導式
セカント	SEC(X)=1/COS(X)
コセカント	CSC(X)=1/SIN(X)
コタンジェント	COT(X)=1/TAN(X)
アークサイン	ARCSIN(X)=ATN(X/SQR(−X*X+1))
アークコサイン	ARCCOS(X)=−ATN(X/SQR(−X*X+1))+1.5708
アークセカント	ARCSEC(X)=ATN(SQR(X*X−1))+(SGN(X)−1)*1.5708
アークコセカント	ARCCSC(X)=ATN(1/SQR(X*X−1))+(SGN(X)−1)*1.5708
アークコタンジェント	ARCCOT(X)=−ATN(X)+1.5708
ハイパーボリックサイン	SINH(X)=(EXP(X)−EXP(−X))/2
ハイパーボリックコサイン	COSH(X)=(EXP(X)+EXP(−X))/2
ハイパーボリックタンジェント	TANH(X)=−EXP(−X)/(EXP(X)+EXP(−X))*2+1
ハイパーボリックセカント	SECH(X)=2/(EXP(X)+EXP(−X))
ハイパーボリックコセカント	CSCH(X)=2/(EXP(X)−EXP(−X))
ハイパーボリックコタンジェント	COTH(X)=EXP(−X)/(EXP(X)−EXP(−X))*2+1
ハイパーボリックアークサイン	ARCSINH(X)=LOG(X+SQR(X*X+1))
ハイパーボリックアークコサイン	ARCCOSH(X)=LOG(X+SQR(X*X−1))
ハイパーボリックアークタンジェント	ARCTANH(X)=LOG((1+X)/(1−X))/2
ハイパーボリックアークセカント	ARCSECH(X)=LOG((SQR(−X*X+1)+1)/X)
ハイパーボリックアークコセカント	ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1)+1)/X)
ハイパーボリックアークコタンジェント	ARCCOTH(X)=LOG((X+1)/(X−1))/2
常用対数	LOG_{10}(X)=LOG(X)/LOG(10)

資料9．メモリマップ

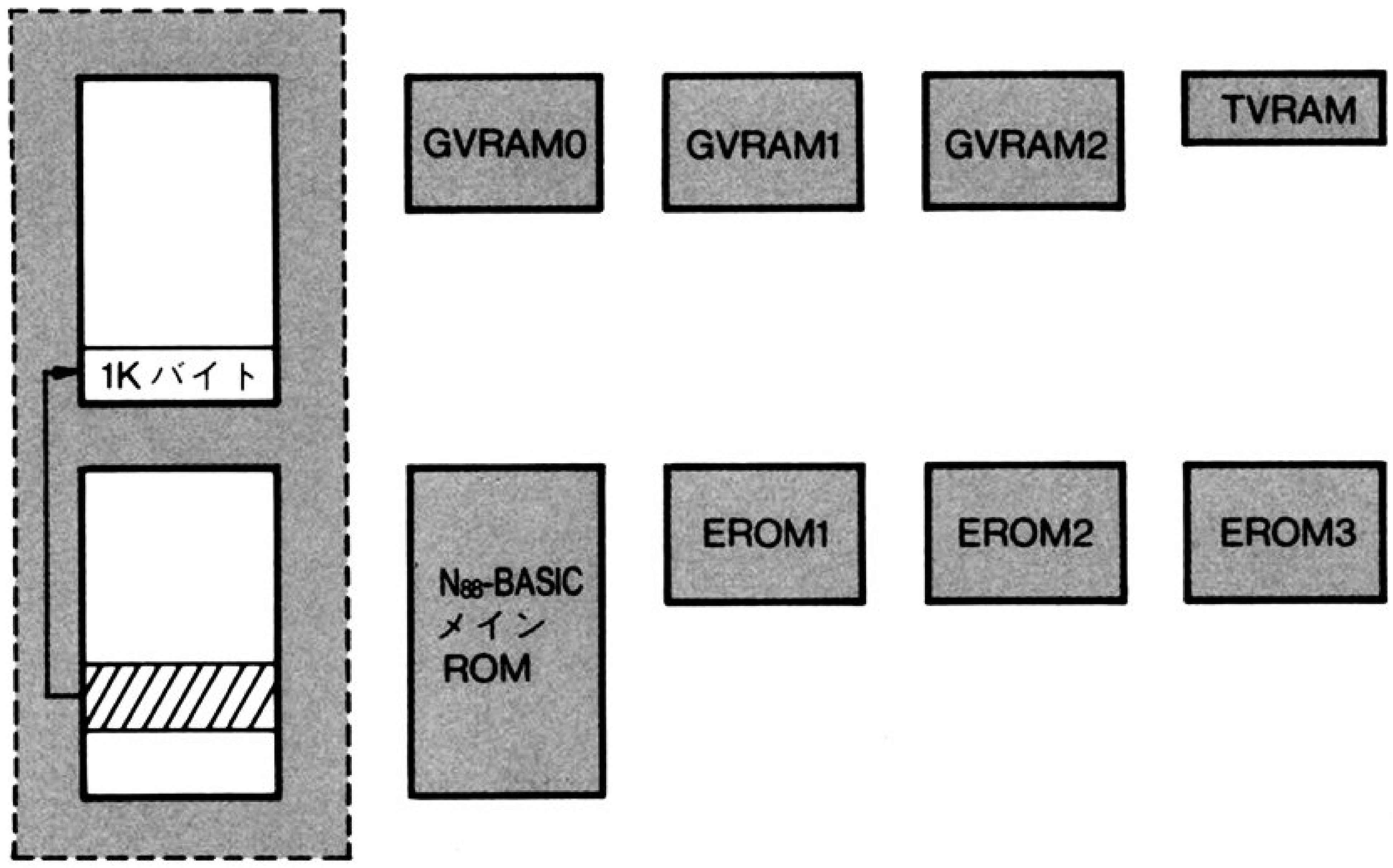
PC-8801MKⅡMRのメモリマップは，次のようになっています．ここで，GVRAMはグラフィックビデオRAM，TVRAMはテキストビデオRAM，EROMはエンハンスト（拡張）ROM，ERAMはエンハンスト（拡張）RAMを意味します．



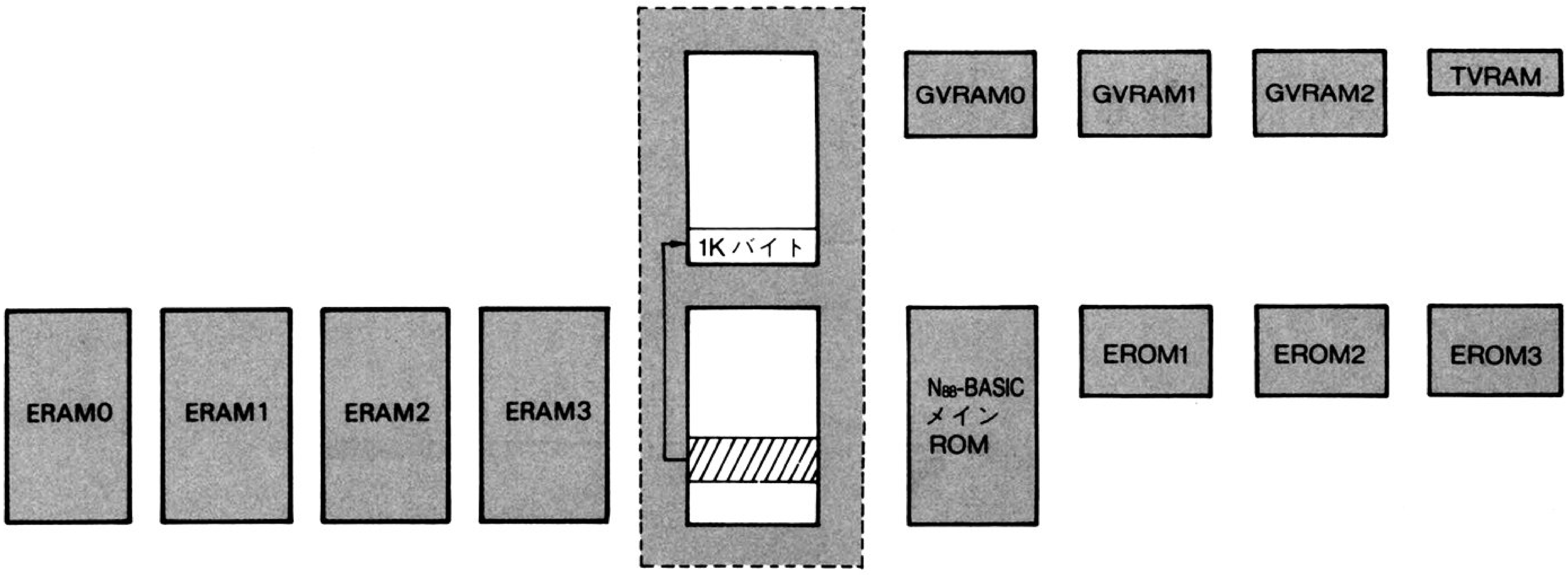
メインCPUのメモリマップ



N88-BASIC V1で使われるメモリ



N88-BASIC V2で使われるメモリ



N88-日本語BASICで使われるメモリ



NEC 日本電気株式会社・日本電気ホームエレクトロニクス株式会社

78117053